

Oracle® Fusion Middleware

Securing Web Services and Managing Policies with Oracle Web Services Manager



14c (14.1.2.0.0)
G12138-01
December 2024

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Oracle Fusion Middleware Securing Web Services and Managing Policies with Oracle Web Services Manager, 14c (14.1.2.0.0)

G12138-01

Copyright © 2019, 2024, Oracle and/or its affiliates.

Primary Author: Panendra Puttachar

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	xlvi
How to Use This Guide	xlvi
Documentation Accessibility	xlix
Related Documents	xlix
Conventions	xlix

What's New in This Guide

New and Changed Features for 14c (14.1.2.0.0)	i
---	---

Part I Introduction to Oracle Web Services

1 Overview of Oracle Web Services Security and Policy Management

1.1 Web Services Security and Policy Management	1-1
1.2 Categories of Oracle Web Services Secured Using OWSM	1-1

2 Using Oracle Web Services Manager with Oracle WebLogic Server

2.1 Installing Oracle Web Services Manager with WebLogic Server	2-1
2.2 Deploying Policy Manager	2-2
2.3 OWSM Configuration with a Domain-Wide Administration Port	2-2
2.3.1 Targeting the Policy Manager to the Administration Server Using the WebLogic Remote Console	2-3
2.3.2 Specifying the policy accessor URL for the Policy Manager on the Administration Server Using Fusion Middleware Control	2-4
2.4 Cross-Component Wiring for Auto-Discovery of Policy Manager	2-4
2.5 About Verifying Service Table Entries and Agent Bindings	2-5
2.5.1 Verifying Service Table Entries and Components Using Fusion Middleware Control	2-5
2.5.2 Verifying Agent Bindings Using Fusion Middleware Control	2-6
2.6 About Modifying the Default User	2-6
2.6.1 Configuring an Authentication Provider	2-7
2.6.2 Configuring the Credential Store Provider	2-7

2.6.3	Configuring the Policy Manager CSF Key for the Domain	2-8
2.6.4	Modifying the User's Group or Role	2-8
2.6.5	Determining that the User Has the Required Role	2-8
2.6.6	Examples of Using OPSS Scripts to Manage Application Roles	2-9

3 Determining Which Predefined Policies to Use for a Web Service

3.1	Security Policy Questionnaire for a Web Service	3-1
3.1.1	Choosing the Right Authentication Policy for a Web Service	3-2
3.1.2	Choosing the Right Confidentiality and Integrity Policy for a Web Service	3-3
3.2	Summary of Predefined Security Policies for a Web Service	3-4
3.2.1	Authentication Only Policies	3-4
3.2.2	Message Protection Only Policies	3-6
3.2.3	Message Protection and Authentication Policies	3-7
3.2.4	Authorization Policies	3-9
3.2.5	WS-Trust Policies	3-9
3.2.6	MTOM Attachment Policies	3-10
3.2.7	Reliable Messaging Policies	3-10
3.2.8	No Behavior Policies	3-11
3.3	OWSM Policies Supported for Java EE Web Services and Clients	3-12
3.4	OWSM Policies Supported for RESTful Web Services and Clients	3-13
3.5	OWSM Policies Supported for Web Services and Clients That Use SOAP Over JMS Transport	3-13
3.6	OWSM Policies Supported for SOA Composite Services and Clients	3-15
3.7	OWSM Policies that Require You to Configure SSL	3-15
3.7.1	List of Policies That Require You to Configure SSL	3-15
3.7.2	List of Templates to Create Policies that Require SSL	3-16
3.7.3	List of Policies That Require You to Configure Two-Way SSL	3-17
3.7.4	List of Templates to Create Policies that Require Two-way SSL	3-17
3.8	OWSM Policies Supported for Identity Context	3-18
3.9	OWSM Policies Supported for WS-SecureConversation	3-19
3.10	OWSM Policies Supported for JCA Adapters	3-19
3.11	OWSM Policies Supported for OES Integration	3-20
3.12	OWSM Policies Are Supported for PII	3-20
3.13	OWSM Policies Supported for Oracle Service Bus	3-20

Part II Attaching and Managing Policies

4 Attaching Policies to Manage and Secure Web Services

4.1	Overview of Policy Attachment	4-1
4.2	Understanding Attaching Policies to Web Services and Clients at Design Time	4-3

4.2.1	About Attaching Policies to Java EE Web Services and Clients at Design Time	4-3
4.2.1.1	Attaching Policies to Java EE Web Services and Clients Using Annotations	4-3
4.2.1.2	Attaching Policies to Java EE Web Service Clients Using Feature Classes	4-5
4.2.1.3	Attaching Policies to Java EE Web Services and Clients Using JDeveloper	4-6
4.2.2	About Attaching Policies to RESTful Web Services and Clients at Design Time	4-6
4.2.2.1	Attaching Policies to RESTful Web Services Using Annotations	4-6
4.2.2.2	Attaching Policies to RESTful Web Service Clients Using Feature Classes	4-8
4.2.2.3	Attaching Policies to RESTful Web Services and Clients Using JDeveloper	4-9
4.2.3	About Attaching Policies to Oracle Infrastructure Web Services and Clients at Design Time	4-9
4.2.3.1	Attaching Policies to Oracle Infrastructure Web Services Using Annotations	4-9
4.2.3.2	Attaching Policies to Oracle Infrastructure Web Service Clients Using Feature Classes	4-10
4.2.3.3	Attaching Policies to Oracle Infrastructure Web Services Using Oracle JDeveloper	4-10
4.3	About Attaching Policies to Web Services and Clients Using Fusion Middleware Control	4-11
4.3.1	Attaching Policies Directly Using Fusion Middleware Control	4-11
4.3.1.1	Attaching Policies Directly to a Single Subject Using Fusion Middleware Control	4-12
4.3.1.2	Attaching Policies Directly to Web Service Clients Using Fusion Middleware Control	4-16
4.3.1.3	Enabling or Disabling Directly Attached Policies Using Fusion Middleware Control	4-20
4.3.1.4	Detaching Directly Attached Policies Using Fusion Middleware Control	4-21
4.3.2	About Attaching Policies Globally Using Fusion Middleware Control	4-21
4.3.2.1	Navigating to the WSM Policy Set Summary Page Using Fusion Middleware Control	4-22
4.3.2.2	Viewing the Configuration of a Policy Set Using Fusion Middleware Control	4-23
4.3.2.3	Creating a Policy Set Using Fusion Middleware Control	4-23
4.3.2.4	Cloning a Policy Set Using Fusion Middleware Control	4-27
4.3.2.5	Editing a Policy Set Using Fusion Middleware Control	4-27
4.3.2.6	Specifying Run-time Constraints in a Policy Set Using Fusion Middleware Control	4-28
4.3.2.7	Enabling and Disabling a Policy Set Using Fusion Middleware Control	4-29
4.3.2.8	Deleting Policy Sets Using Fusion Middleware Control	4-30
4.3.3	Viewing Policies Attached to a Web Service Using Fusion Middleware Control	4-30
4.3.4	Validating Policy Attachments	4-31
4.3.5	About Validating a Policy Set	4-33
4.4	About Attaching Policies to Web Services and Clients Using WLST	4-33
4.4.1	Viewing Available Policies Using WLST	4-34
4.4.2	About Attaching Policies Directly to Java EE Web Services and Clients Using WLST	4-34
4.4.2.1	Viewing the Policies That Are Attached to a Java EE Web Service	4-35
4.4.2.2	Viewing the Policies That Are Attached to a Java EE Web Service Client	4-36

4.4.2.3	Attaching Policies Directly to a Java EE Web Service Using WLST	4-36
4.4.2.4	Attaching Policies Directly to Java EE Web Service Clients Using WLST	4-37
4.4.2.5	Detaching Directly Attached Policies from Java EE Web Service and Clients Using WLST	4-39
4.4.2.6	Enabling and Disabling Web Service Client Policies Using WLST	4-41
4.4.3	About Attaching Policies Directly to RESTful and Oracle Infrastructure Web Services and Clients Using WLST	4-41
4.4.3.1	Identifying and Selecting the Policy Subject Using WLST	4-42
4.4.3.2	Attaching Policies Directly Using WLST	4-43
4.4.3.3	Enabling and Disabling Directly Attached Policies Using WLST	4-45
4.4.3.4	About Detaching Directly Attached Policies Using WLST	4-46
4.4.4	About Attaching Policies Globally Using WLST	4-48
4.4.4.1	Viewing a List of Policy Sets	4-49
4.4.4.2	Displaying the Configuration of a Policy Set	4-49
4.4.4.3	Managing Sessions Using WLST	4-50
4.4.4.4	Creating a New Policy Set Using WLST	4-50
4.4.4.5	Cloning a Policy Set using WLST	4-53
4.4.4.6	Editing a Policy Set	4-55
4.4.4.7	Validating a Policy Set	4-57
4.4.4.8	Enabling and Disabling a Policy Set	4-57
4.4.4.9	Deleting Policy Sets Using WLST	4-58
4.4.4.10	Specifying Runtime Constraints in Policy Sets Using WLST	4-60
4.4.5	Viewing Policies Attached to a Web Service with WLST	4-61
4.4.6	Displaying the Effective Policy Set Using WLST	4-62
4.5	About Attaching Policies to Servlet Applications	4-64
4.5.1	Attaching Policies Directly to Servlet Applications	4-64
4.5.2	Attaching Policies Globally to Servlet Applications	4-67
4.6	About Securing the URI patterns for Resources in RESTful Web Services	4-69
4.6.1	Example Scenario: Creating Policies to Secure URI patterns	4-69
4.6.1.1	Creating a Policy Set to Secure the Application	4-70
4.6.1.2	Creating a Policy Set to Secure the Module in an Application	4-71
4.6.1.3	Creating Policy Sets to Secure Multiple Modules with Different Policies in the Application	4-72
4.6.1.4	Creating a Policy Set to Secure the Paths for a Module in the Service	4-73
4.6.1.5	Creating Policy Sets to Grant Anonymous Access for a Specific Path in a Secured Application	4-74
4.6.1.6	Creating a Policy Set to Secure the HTTP Method in the Application	4-76
4.7	Run-time Constraints in Policy Sets	4-77
4.8	About Defining the Type and Scope of Resources for Globally Attached Policies	4-79
4.8.1	Defining the Resource Type	4-80
4.8.2	Defining the Resource Scope	4-80
4.8.3	Determining the Namespace for a Web Service	4-82
4.8.4	Examples of Creating Policy Sets Using Different Resource Types and Scopes	4-83

4.9	Migrating Direct Policy Attachments to Global Policy Attachments	4-84
4.10	Disabling a Globally Attached Policy	4-86
4.11	Specifying the Priority of a Policy Attachment	4-87
4.12	Managing Endpoint Configuration Properties Using Fusion Middleware Control	4-88
4.13	Determining the Secure Status of an Endpoint	4-88
4.14	How the Effective Set of Policies is Calculated	4-90
4.15	Determining the Source of Policy Attachments	4-93

5 Overriding Policy Configuration Properties

5.1	Overview of Policy Configuration Overrides	5-1
5.2	Scope of Predefined Configuration Properties	5-2
5.3	About Overriding Client Policy Configuration Properties at Design Time	5-3
5.3.1	Java EE Web Services	5-3
5.3.2	RESTful Web Services	5-3
5.3.3	Understanding Oracle Infrastructure Web Services	5-3
5.3.3.1	Client Policy Configuration Properties That Can Be Overridden at Design Time	5-4
5.3.3.2	Example for Overriding the Client Policy Configuration Properties for Keystore, Username, and Password Using RequestContext	5-7
5.3.3.3	Example for Overriding the RESTful Web Service Client Policy Configuration Properties for the Username and Password	5-8
5.4	About Overriding Policy Configuration Properties Using Fusion Middleware Control	5-9
5.4.1	Overriding Configuration Properties at the Domain Level (Defining the Default Value)	5-9
5.4.2	Overriding Configuration Properties for Directly Attached Service Policies Using Fusion Middleware Control	5-10
5.4.3	Overriding Configuration Properties at the Web Service Client Application Level Using Fusion Middleware Control	5-11
5.4.4	Overriding Configuration Properties for Globally Attached Policies Using Fusion Middleware Control	5-12
5.5	About Overriding Policy Configuration Properties Using WLST	5-13
5.5.1	Overriding Configuration Properties for Directly Attached Service Policies Using WLST	5-14
5.5.2	Overriding Configuration Properties at the Web Service Client Application Using WLST	5-15
5.5.3	Overriding Configuration Properties for Globally Attached Policies Using WLST	5-16
5.6	About Configuring User-Defined Properties for Web Service and Client Policies Using Fusion Middleware Control	5-18
5.6.1	Scope of User-Defined Configuration Properties	5-18
5.6.2	Adding a User-Defined Configuration Property	5-19
5.6.3	Editing a User-Defined Configuration Property	5-19
5.6.4	Deleting a User-Defined Configuration Property	5-19

6 Managing Web Service Policies with Fusion Middleware Control

6.1	Overview of Web Services Policy Management	6-1
6.2	Managing Web Service Policies	6-1
6.2.1	Navigating to the WSM Policies Page	6-2
6.2.2	Searching for Policies in the WSM Policies Page	6-2
6.2.2.1	Using Advanced Search	6-3
6.2.2.2	Using the Query by Example Filter	6-3
6.2.3	Viewing the Details of a Web Service Policy	6-4
6.2.4	Creating and Editing Web Service Policies	6-6
6.2.4.1	Creating a New Web Service Policy	6-6
6.2.4.2	Cloning a Web Service Policy	6-8
6.2.4.3	Creating Custom Policies	6-9
6.2.4.4	Editing a Web Service Policy	6-9
6.2.5	Using Local Optimization with OWSM Policies (SOA Composites)	6-10
6.2.5.1	Viewing the Default Local Optimization Setting in OWSM Policies	6-10
6.2.5.2	Controlling When Local Optimization is Used	6-11
6.2.6	Generating Client Policies from a WSDL	6-11
6.2.7	Adding Assertions to a Policy	6-14
6.2.8	Adding an OR Group to a Policy	6-15
6.2.9	Importing Web Service Policies	6-16
6.2.10	Exporting Web Service Policies	6-18
6.2.11	Versioning Web Service Policies	6-18
6.2.11.1	Viewing the Version History of a Web Service Policy	6-19
6.2.11.2	Changing the Current Version of a Policy	6-20
6.2.11.3	Deleting Versions of a Web Service Policy	6-20
6.2.11.4	Exporting a Version of a Policy	6-20
6.2.12	Deleting a Web Service Policy	6-21
6.3	Validating Web Service Policies	6-21
6.4	Managing Policy Assertion Templates	6-22
6.4.1	About Navigating to the Assertion Templates Page	6-23
6.4.2	Understanding Search Options on the Assertion Templates Page	6-23
6.4.2.1	Searching for an Assertion Template Using Advanced Search	6-23
6.4.2.2	Searching for an Assertion Template Using the Query by Example Filter	6-24
6.4.3	Viewing the Details of an Assertion Template	6-25
6.4.4	Naming Conventions for Assertion Templates	6-26
6.4.5	Cloning an Assertion Template	6-27
6.4.6	Editing an Assertion Template	6-28
6.4.7	Editing the Configuration Properties in an Assertion Template	6-28
6.4.8	Exporting an Assertion Template	6-29

6.4.9	Importing an Assertion Template	6-30
6.4.10	Deleting an Assertion Template	6-31
6.5	Managing Policies and Assertions	6-31
6.5.1	Enabling or Disabling a Policy for all Policy Subjects	6-31
6.5.2	Enabling or Disabling Assertions Within a Policy	6-32
6.6	Analyzing Policy Usage	6-33
6.7	About Advertising Policy Assertions	6-34
6.8	About Advertising WS-Policy and WS-SecurityPolicy Versions	6-35

Part III Securing Web Services

7 Configuring Message Protection for Web Services

7.1	Overview of Message Protection Configuration for Web Services	7-1
7.2	Overview of Configuring Keystores for Message Protection	7-2
7.2.1	Understanding OPSS Keystore Service for Message Protection	7-2
7.2.1.1	Configuring Message Protection Using the OPSS Keystore Service	7-3
7.2.1.2	Migrating a JKS Keystore Into the KSS Keystore	7-5
7.2.1.3	Importing Certificates Into the KSS Keystore	7-6
7.2.1.4	Overriding keystore.sig.csf.key and keystore.enc.csf.key Attributes	7-6
7.2.1.5	Renewing or Regenerating the Expiring Certificates or Keys	7-7
7.2.2	Understanding Java Keystore for Message Protection	7-7
7.2.2.1	Generating Private Keys and Creating the Java Keystore	7-7
7.2.2.2	Obtaining a Trusted Certificate and Importing it into the Keystore	7-9
7.2.2.3	Configuring the OWSM Keystore	7-10
7.2.3	Adding Keys and User Credentials to Configure the Credential Store	7-10
7.2.3.1	Adding Keys and User Credentials to the Credential Store Using Fusion Middleware Control	7-12
7.2.3.2	Adding Keys and User Credentials to the Credential Store Using WLST	7-15
7.3	About Creating an Application-level Credential Map	7-15
7.3.1	How CSF Keys Are Retrieved from an Application-level Credential Map	7-15
7.3.2	About Permissions to Access an Application-level Credential Map	7-16
7.3.2.1	Configuring the csf.map Property Override	7-16
7.3.2.2	About Granting CredentialAccessPermission to wsm-agent-core.jar	7-17
7.3.2.3	About Grant WSIdentityPermission to wsm-agent-core.jar	7-18
7.3.2.4	Example of Granting Permission for Application-level In system-jazn-data.xml	7-19
7.3.3	Policies that Can Be Used to Access an Application-level CSF Map	7-20
7.4	Understanding Service Identity Certificate Extensions	7-22
7.4.1	Ignoring the Service Identity Certificate Extension From the Client	7-23
7.4.2	Ignoring Hostname Verification from the Client	7-24
7.5	Caching the Nonce with Oracle Coherence	7-24

7.5.1	Caching the Nonce Where There Are No Managed Coherence Servers	7-25
7.5.1.1	Understanding Coherence Cluster Topology Where There Are No Managed Coherence Servers	7-25
7.5.1.2	Configuring the Standard Topology Using Fusion Middleware Configuration Wizard	7-26
7.5.2	Caching the Nonce for Storage-Disabled WebLogic Servers and Storage-Enabled Managed Coherence Servers	7-26
7.5.2.1	Understanding Coherence Cluster Topology For Storage-Disabled WebLogic Servers and Storage-Enabled Managed Coherence Servers	7-26
7.5.2.2	Configuring the Cluster Topology Using Fusion Middleware Configuration Wizard	7-27
7.6	About Configuring Partial Encryption with Fusion Middleware Control	7-28
7.6.1	Configuring Partial Encryption Using Fusion Middleware Control	7-28
7.6.2	Securing SwA Attachments	7-30

8 Protecting Personally Identifiable Information

8.1	Main Steps in Protecting PII Information	8-1
8.1.1	Approach to Follow to Determine What PII Data to Protect	8-1
8.1.2	Composing the XPath Expressions to Protect the PII Data	8-2
8.1.3	Configuring the PII Encryption Key	8-2
8.1.4	Attaching the pii_security_policy Policy	8-2
8.1.5	Attaching the pii_security_policy to SOA Composite	8-3
8.1.6	Attaching the pii_security_policy to Oracle Service Bus	8-5
8.1.7	Attaching the pii_security_policy to JCA Binding	8-5
8.2	Overriding the pii_security_policy Attributes Using WLST	8-5
8.3	Decrypting PII Using API	8-6

9 Configuring Transport-Level Security (SSL)

9.1	About Configuring Keystores for SSL	9-1
9.1.1	Understanding KSS Keystore Configuration on WebLogic Server	9-1
9.1.1.1	Configuring the OPSS Keystore Service for Demo Identity and Trust	9-2
9.1.1.2	Recreating the OPSS Keystore Service for Demo Identity and Trust	9-2
9.1.1.3	Configuring the OPSS Keystore Service for Custom Identity and Trust	9-5
9.1.2	Configuring a JKS Keystore on WebLogic Server	9-7
9.1.3	Configuring Synchronization of JKS Keystore File on Cluster	9-9
9.2	Configuring One-Way SSL on WebLogic Server	9-10
9.3	Configuring Two-Way SSL on WebLogic Server	9-11
9.4	Configuring One-Way SSL for a Web Service Client	9-11
9.5	Configuring Two-Way SSL for a Web Service Client	9-12
9.6	Understanding SSL Configuration on Oracle HTTP Server	9-13
9.6.1	Configuring One-Way SSL on Oracle HTTP Server	9-14

10 Configuring Authorization Using Oracle Web Services Manager

10.1	Overview of Authorization	10-1
10.2	Determining Which Resources to Protect	10-2
10.3	Determining Authorization Permissions	10-3
10.4	Determining the OPSS Resource Name	10-5
10.5	About Configuring Fine-Grained Authorization Using Oracle Entitlements Server	10-5
10.5.1	Prerequisites for Configuring OES Integration	10-6
10.5.2	Understanding Attributes for Obligations	10-6
10.5.3	About Configuring OES Policies For Fine-Grained Authorization	10-7
10.5.3.1	Configuring the OES Resource for Masking	10-7
10.5.3.2	Creating Authorization Policy to Return Obligations	10-8
10.5.3.3	Creating Actual OES Authorization Policy for Coarse-Grained Authorization	10-11
10.5.4	About Configuring OES Policies For Coarse-Grained Authorization	10-13
10.5.4.1	Configuring the OES Resource for Coarse-Grained Authorization	10-13
10.5.4.2	Creating Actual OES Authorization Policy for Fine-Grained Authorization	10-14
10.5.5	About Configuring OES Policy For Masking	10-16
10.5.5.1	Configuring the OES Resource	10-16
10.5.5.2	Creating Masking Policy to Return Obligations	10-17
10.5.5.3	Creating Actual OES Masking Policy	10-17
10.5.6	Understanding How to Attach OWSM OES Policy	10-18
10.5.6.1	Attaching the OWSM OES Policy	10-18
10.5.6.2	Configuration Properties and Overrides	10-19
10.6	Configuring the Oracle HTTP Server to Specify the Request Origin	10-21
10.7	Using OAuth 2.0 with Oracle Web Services Manager	10-21
10.7.1	About OAuth2 with Oracle Web Services Manager	10-22
10.7.1.1	Understanding 2-legged OAuth2	10-22
10.7.1.2	Supported Authorization Grant Types in 2-Legged Authorization	10-23
10.7.1.3	How Client Credentials Are Determined in 2-Legged Authorization	10-24
10.7.1.4	Relationship of User Credentials, Client Credentials, and Subject in 2-Legged Authorization	10-24
10.7.2	About OWSM OAuth2 Client API	10-25
10.7.2.1	Configuring OWSM OAuth2 Client Credentials	10-25
10.7.3	Configuring OAuth2 for Use With Oracle Web Services Manager Policies	10-28
10.7.4	Enabling User Assertion by Username/Password in OAuth Policy	10-28
10.7.5	About OAuth Based Anonymous User Authentication	10-29
10.7.6	About Multitenant Support for OAuth Token Validation	10-30
10.7.7	About Proxy Configuration for Outbound OAuth 2.0 on the Client Application	10-30
10.7.7.1	Enabling Proxy for OAuth2 Server Token Endpoints	10-31
10.7.7.2	Managing the Proxy Server Exclusion Lists	10-33

10.7.8	About Support for OAuth 2.0 Tokens with Application Defined Scopes	10-36
10.8	Understanding OWSM integration with 3rd Party Servers	10-38
10.8.1	About OWSM Integration with Twitter OAuth server	10-38
10.8.1.1	Prerequisites for configuring Twitter account	10-39
10.8.1.2	Configuring Twitter account for integration with OWSM	10-39
10.8.2	About OWSM Integration with Google OAUTH2 Server	10-41
10.8.2.1	Creating a Google Project Using the Google API Console	10-41
10.8.2.2	Configuring Google Service Account	10-41
10.8.2.3	Enabling Google API	10-42
10.8.2.4	Configuring the Keystore for the Client Application	10-42
10.8.2.5	Verifying Integration with Google OAuth Endpoints Using JDeveloper	10-43

11 Configuring Authentication Using Oracle Web Services Manager

11.1	Overview of Authentication Configuration	11-1
11.2	Supported Authentication Providers in WebLogic Server	11-2
11.3	About Configuring Digest Authentication	11-3
11.3.1	Prerequisites for Configuring Digest Authentication	11-3
11.3.2	Configuring the Default Authenticator and Identity Asserter	11-3
11.3.3	Attaching a Policy and Enabling Digest Authentication	11-4
11.4	About SAML Configuration	11-5
11.4.1	Overview of Flow of SAML Token Validation	11-5
11.4.1.1	Validating a SAML Assertion	11-6
11.4.1.2	Use Cases for SAML Token Validation	11-6
11.4.2	Configuring SAML Web Service Client at Design Time	11-7
11.4.3	Including User Attributes in the Assertion	11-8
11.4.4	Including User Roles in the Assertion	11-9
11.4.5	Understanding the Configuration of Oracle Platform Security Services (OPSS) for SAML Policies	11-9
11.4.6	Adding an Additional SAML Assertion Issuer Name	11-10
11.4.7	About SAML Web Service Client Configuration for Identity Switching	11-11
11.4.7.1	Understanding Identity Switching Use Case Scenarios	11-12
11.4.7.2	Setting the javax.xml.ws.security.auth.username Property	11-13
11.4.7.3	Setting the Permission Using WSIdentityPermission	11-13
11.4.8	Understanding Trusted Issuers and Trusted Distinguished Names List for SAML Signing Certificates	11-14
11.4.9	Understanding How to Use Anonymous Users with SAML Policies	11-15
11.5	About Propagating Identity Context with OWSM	11-15
11.5.1	Overview of Identity Context	11-15
11.5.2	Propagating Identity Context Using SAML Policies	11-16
11.5.3	Configuring Identity Context Propagation: Main Steps	11-16
11.6	Understanding Kerberos Token Configuration	11-18
11.6.1	About MIT Kerberos	11-18

11.6.1.1	Initializing and Starting the MIT Kerberos KDC	11-18
11.6.1.2	Creating Principals	11-19
11.6.1.3	Configuring the Web Service Client to Use the Correct KDC	11-19
11.6.2	About Using Microsoft Active Directory with Key Distribution Center	11-20
11.6.2.1	Web Service Client Set Up Tasks	11-21
11.6.2.2	Setting Up the Web Service	11-22
11.6.3	Setting the Service Principal Name In the Web Service Client	11-22
11.6.4	Configuring the Web Service to Use the Correct KDC	11-23
11.6.5	About Using the Correct Keytab File in Enterprise Manager	11-23
11.6.5.1	Extracting the Keytab File	11-23
11.6.5.2	Exporting the Keytab File	11-23
11.6.5.3	Modifying the krb5 Login Module to use the Keytab File	11-24
11.6.6	Authenticating the User Corresponding to the Service Principal	11-24
11.6.7	Creating a Ticket Cache for the Web Service Client	11-24
11.6.8	Kerberos Configuration Over SSL	11-25
11.6.9	Kerberos Configuration with SPNEGO Negotiation	11-25
11.6.10	About Configuration of Credential Delegation	11-25
11.6.10.1	Configuring Credential Delegation in Kerberos	11-26
11.6.10.2	Configuring Delegation Permission to OWSM	11-26
11.6.10.3	Enabling Credential Delegation in the Client Policy and the Service Policy	11-27
11.6.10.4	Configuring Credential Delegation for a Service in Active Directory	11-27
11.6.10.5	Configuring Credential Delegation With An Example	11-27
11.7	About WS-Trust Configuration	11-28
11.7.1	Overview of Web Services WS-Trust	11-29
11.7.1.1	Understanding the Mechanism to Obtain STS Configuration	11-30
11.7.1.2	Understanding the Token Types Exchanged	11-31
11.7.2	Supported STS Servers	11-31
11.7.3	Understanding Token Lifetime and Token Caching	11-32
11.7.4	Setting Up Automatic Policy Configuration for STS	11-32
11.7.4.1	Understanding the Requirements for Automatic Policy Configuration	11-32
11.7.4.2	Main Steps in Setting Up Automatic Policy Configuration	11-33
11.7.4.3	Manually Configuring the STS Config Policy From the Web Service Client: Main Steps	11-35
11.7.5	About Configuring Web Services Federation	11-37
11.7.5.1	Configuring a Web Service for Web Services Federation	11-37
11.7.5.2	About Configuring a Web Client for Web Services Federation	11-37
11.7.5.3	Configuring an STS for Web Services Federation	11-39
11.7.6	Overview of SAML Holder of Key and SAML Bearer as Issued Tokens	11-39
11.7.6.1	Determining the Proof Key for SAML HOK Only	11-41
11.7.7	Understanding SAML Sender Vouches as Issued Tokens	11-42
11.7.8	Overview of On Behalf Of Use Cases	11-43
11.7.9	Programmatically Overriding Policy Configuration for WS-Trust Client Policies	11-43

11.7.10	Overview of JWK Document Trust Configuration	11-45
11.7.11	About IDCS Discovery Service	11-46
11.7.12	About Token Audience Configuration	11-46

12 Configuring Secure Conversation Using Oracle Web Services Manager

12.1	Overview of Web Services Secure Conversation Language Specification	12-1
12.2	About Configuring Secure Conversation	12-1
12.2.1	Configuring Secure Conversation Using Fusion Middleware Control	12-2
12.2.2	Configuring Secure Conversation Using WLST	12-3
12.3	Attaching a Secure Conversation Policy at Design Time	12-4
12.4	About Configuring Persistence	12-4
12.4.1	Overview of Persistence	12-4
12.4.2	Configuring Persistence for a Web Service	12-5
12.4.3	Configuring Persistence for a Client	12-6
12.5	Understanding Secure Conversation Sessions	12-6

13 Integrating Hardware with Oracle Web Services Manager

13.1	Using Hardware Security Modules With OWSM	13-1
13.1.1	Understanding SafeNet Luna SA With OWSM for Key Storage	13-1
13.1.2	About Installing and Configuring the Luna SA HSM Client	13-2
13.1.3	Configuring the JRE Used By OWSM	13-2
13.1.4	Logging On to Luna SA	13-3
13.1.5	Copying Keys and Certificates to Luna SA	13-3
13.1.6	About Configuring OWSM to Use Luna SA	13-4
13.2	About Configuring OWSM for Oracle SPARC T5 and SPARC T4 Cryptographic Acceleration	13-4
13.2.1	Terms You Need to Understand	13-4
13.2.2	Overview of Oracle SPARC T5 and SPARC T4 Hardware Assisted Cryptographic Acceleration	13-5
13.2.3	Configuring Transport-Level Security for Cryptographic Acceleration	13-6
13.2.4	Configuring Message-level Security for Cryptographic Acceleration	13-7
13.2.5	Additional Reading for Cryptographic Acceleration	13-8

Part IV Managing and Troubleshooting Oracle Web Services Manager

14 Managing Oracle Web Services Manager Domain Configuration

14.1	Overview of OWSM Domain Configuration	14-1
14.2	Navigating to the WSM Domain Configuration Page	14-2
14.3	Viewing the General OWSM Domain Configuration Using Fusion Middleware Control	14-2

14.4	Configuring Domain-Level Authentication Using Fusion Middleware Control	14-3
14.4.1	SAML Trusted Issuers and DN Lists Using Fusion Middleware Control	14-3
14.4.1.1	Overview of SAML Trusted Issuers and DN Lists	14-3
14.4.1.2	Adding SAML Issuers and Defining a Trusted DN List Using Fusion Middleware Control	14-4
14.4.1.3	Deleting Trusted Issuers, DNSs, or DN Lists Using Fusion Middleware Control	14-5
14.4.2	Configuring JWT Trusted Issuers and DN Lists Using Fusion Middleware Control	14-5
14.4.2.1	About JWT Trusted Issuers and DN Lists	14-6
14.4.2.2	Adding JWT Issuers and Defining a Trusted DN List Using Fusion Middleware Control	14-6
14.4.3	Configuring Token Attribute Rules for Trusted Issuers Using Fusion Middleware Control	14-7
14.4.4	Configuring the Lifetime for the Issued Token Using Fusion Middleware Control	14-8
14.4.5	Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control	14-8
14.4.6	Configuring the Kerberos Login Module	14-9
14.4.7	Configuring Subject Properties Using Fusion Middleware Control	14-10
14.4.8	Configuring the X509 Login Module Using Fusion Middleware Control	14-11
14.4.9	Creating Custom Login Modules	14-11
14.5	Domain-Level Message Security Configuration Using Fusion Middleware Control	14-12
14.5.1	OWSM Keystore Configuration Using Fusion Middleware Control	14-13
14.5.1.1	Configuring OWSM to Use the KSS Keystore	14-13
14.5.1.2	Configuring OWSM to Use the JKS Keystore	14-15
14.5.1.3	Configuring OWSM to Use HSM Keystores	14-16
14.5.1.4	Configuring OWSM to Use the PKCS11 Keystore	14-17
14.5.2	Configuring Security Policy Enforcement Using Fusion Middleware Control	14-19
14.5.3	Configuring Identity Extension Properties Using Fusion Middleware Control	14-21
14.5.4	Secure Conversation Configuration for the Domain Using Fusion Middleware Control	14-22
14.5.4.1	About Secure Conversation	14-22
14.5.4.2	Configuring Secure Conversation with Fusion Middleware Control	14-23
14.6	OWSM Policy Access Configuration Using Fusion Middleware Control	14-23
14.6.1	Understanding Configuring the Policy Manager Connection Using Fusion Middleware Control	14-24
14.6.1.1	About Auto-Discovery and Connecting to the Policy Manager	14-24
14.6.1.2	Configuring a Connection to the Policy Manager Using Fusion Middleware Control	14-24
14.6.2	About Refreshing Configuration Cache in OWSM Manually by using Fusion Middleware Control	14-26
14.6.2.1	Disabling Automatic Refresh Option in OWSM by using Fusion Middleware Control	14-26
14.6.2.2	Checking Status of Automatic Refresh in OWSM by using Fusion Middleware Control	14-27

14.6.2.3	Refreshing the OWSM Cache Manually by using Fusion Middleware Control	14-28
14.6.3	Configuring SSL for the Policy Manager Connection Using Fusion Middleware Control	14-29
14.6.4	High Availability Configuration and Cache Management Using Fusion Middleware Control	14-30
14.6.4.1	About High Availability and Cache Management	14-30
14.6.4.2	Configuring High Availability and Managing the Cache Using Fusion Middleware Control	14-31
14.7	About Managing OWSM Domain Configuration Properties Using WLST	14-31
14.7.1	Viewing OWSM Domain Configuration Using WLST	14-32
14.7.2	Setting OWSM Domain Configuration Properties Using the setWSMConfiguration Command	14-32
14.8	Configuring Domain-Level Authentication Using WLST	14-34
14.8.1	Configuring SAML and JWT Trusted Issuers, DN Lists, and Token Attribute Rules Using WLST	14-34
14.8.2	Deleting a Token Issuer Trust Document Using WLST	14-39
14.8.3	Configuring the Lifetime for the Issued Token Using WLST	14-40
14.8.4	Configuring Subject Properties Using WLST	14-40
14.8.5	Configuring the SAML and SAML2 Login Modules Using WLST	14-41
14.8.6	Configuring the Kerberos Login Module Using WLST	14-42
14.8.7	Configuring the X509 Login Module Using WLST	14-43
14.8.8	Configuring Custom Login Modules Using WLST	14-44
14.9	About Configuring Domain-Level Message Security Using WLST	14-44
14.9.1	Configuring the OWSM Keystore Using WLST	14-45
14.9.2	Configuring Security Policy Enforcement Using WLST	14-47
14.9.3	Configuring Identity Extension Properties Using WLST	14-48
14.9.4	Configuring Secure Conversation for the Domain Using WLST	14-49
14.10	About Configuring Policy Access Using WLST	14-50
14.10.1	Configuring the Policy Manager Connection Using WLST	14-50
14.10.2	Updating Bootstrap Configuration Properties Using the setWSMBootstrapConfig Command	14-52
14.10.3	About Refreshing Configuration Cache in OWSM Manually by using WLST	14-53
14.10.3.1	Disabling Automatic Refresh Option in OWSM by using WLST	14-54
14.10.3.2	Checking Status of Automatic Refresh in OWSM by using WLST	14-54
14.10.3.3	Refreshing the OWSM Cache Manually by using WLST	14-54
14.10.4	Configuring High Availability and Cache Management Using WLST	14-55

15 Managing the Oracle Web Services Manager Repository

15.1	Overview of OWSM Repository	15-1
15.2	Registering an OWSM Repository	15-1
15.3	Understanding the Different Mechanisms for Importing and Exporting Policies	15-2
15.4	About Importing and Exporting Documents in the Repository Using WLST	15-3

15.4.1	Exporting Documents from the Repository Using WLST	15-3
15.4.2	Exporting Application Metadata from the Repository Using WLST	15-4
15.4.3	Importing Documents into the Repository Using WLST	15-5
15.5	Exporting Policies from the OWSM Repository for Use in JDeveloper	15-6
15.6	About Patching Policies in the Repository	15-6
15.7	Creating Back Up and Restoring the OWSM Repository	15-7
15.8	Upgrading the OWSM Repository	15-8
15.9	Rebuilding the OWSM Repository	15-9
15.10	Configuring Multiple Domains for a Single OWSM Repository Instance	15-9

16 Diagnosing Problems with Oracle Web Services Manager

16.1	Diagnosing Policy Manager Problems Using the OWSM Policy Manager Page	16-1
16.2	Overview of Common Problems with Oracle Web Services Manager	16-3
16.2.1	Overview of Common Policy Manager Connection Problems	16-3
16.2.1.1	Understanding Common Policy Manager Connection Problems	16-3
16.2.1.2	Solving Policy Manager Connection Problems	16-5
16.2.2	Overview of Key Store or Credential Store Errors After an Application Invokes a Web Service	16-6
16.2.2.1	Understanding Common Key or Credential Store Errors	16-6
16.2.2.2	Solving Key and Credential Store Problems	16-6
16.2.3	Overview of Trust Certificate Error After Application Invokes a Web Service	16-8
16.2.3.1	Understanding a Trust Certificate Error	16-8
16.2.3.2	Solving Trust Certificate Problems	16-8
16.2.4	About Troubleshooting SAML Assertion Errors During Identity Propagation	16-9
16.2.4.1	Understanding Common SAML Assertion Problems	16-9
16.2.4.2	Troubleshooting SAML Assertion Problems	16-9
16.2.5	Overview of Policy Access Problems After an Application Invokes a Web Service	16-10
16.2.5.1	Understanding Common Policy Access Problems	16-10
16.2.5.2	Solving Common Policy Access Problems	16-10
16.2.6	Overview of Problems Accessing Users in the Credential Store	16-11
16.2.6.1	Understanding Common User Access Problems	16-11
16.2.6.2	Solving User Access Problems	16-11
16.2.7	Overview of Common User Authorization Problems After an Application Invokes a Web Service	16-12
16.2.7.1	Understanding Common User Authorization Problems	16-12
16.2.7.2	Solving a User Authorization Problem	16-12
16.2.8	Overview of Timestamp Errors After an Application Invokes a Web Service	16-13
16.2.8.1	Understanding the Causes a Timestamp or clockSkew Error	16-13
16.2.8.2	Solving Timestamp or clockSkew Errors	16-13
16.2.9	Overview of Multiple Authentication Security Policy Errors After an Application Invokes a Web Service	16-14
16.2.9.1	Understanding Common Multiple Authentication Security Policy Errors	16-14

16.2.9.2	Solving Multiple Authentication Security Policy Errors	16-15
16.3	Overview of Policy Attachment Issues Using WLST	16-16
16.3.1	Understanding the Use of listWSMPolicySubjects Command to Identify Policy Attachment Issues	16-16
16.3.2	Viewing a Sample Configuration Output with Globally and Directly Attached Policies	16-17
16.3.3	Viewing a Sample Valid Configuration Output with Directly Attached Policies Only	16-18
16.4	About Diagnosing Problems With a Domain Configuration Using WLST	16-19
16.4.1	Understanding the Use of checkWSMStatus Command to Identify Domain Configuration Issues	16-19
16.4.2	Viewing checkWSMStatus Output Showing Status	16-20
16.4.3	Viewing checkWSMStatus Output Showing Credential Store Failure	16-21
16.4.4	Viewing checkWSMStatus Output With OAuth2 Global Policy Set Configured	16-22
16.4.5	Viewing checkWSMStatus Output With OAuth2 Global Policy Set Not Configured	16-22
16.5	Common Oracle Web Services Manager Exceptions for WS-Trust Use Cases	16-23
16.6	Managing third party server integration with OWSM	16-24

Part V Oracle Web Services Manager Predefined Policies and Assertions Templates

17 Oracle Web Services Manager Predefined Policies

17.1	Addressing Policies	17-2
17.2	Atomic Transaction Policies	17-2
17.3	Configuration Policies	17-2
17.4	Management Policies	17-4
17.5	MTOM Policies	17-5
17.6	Reliable Messaging Policies	17-5
17.7	Security Policies-Authentication Only	17-5
17.8	Security Policies-Authorization Only	17-8
17.9	Security Policies-Message Protection Only	17-8
17.10	Security Policies-Messages Protection and Authentication	17-9
17.11	Security Policies-Sha256 Only	17-14
17.12	Security Policies—Oracle Entitlements Server	17-15
17.13	SOAP Over JMS Transport Policies	17-15
17.14	oracle/wsaddr_policy	17-16
17.15	oracle/no_addressing_policy	17-16
17.16	oracle/atomic_transaction_policy	17-17
17.17	oracle/no_atomic_transaction_policy	17-19
17.18	oracle/async_web_service_policy	17-19
17.19	oracle/cache_binary_content_policy	17-21
17.20	oracle/fast_infoset_client_policy	17-22

17.21	oracle/fast_infoset_service_policy	17-23
17.22	oracle/max_request_size_policy	17-24
17.23	oracle/mex_request_processing_service_policy	17-25
17.24	oracle/mtom_encode_fault_service_policy	17-26
17.25	oracle/no_async_web_service_policy	17-27
17.26	oracle/no_cache_binary_content_policy	17-28
17.27	oracle/no_fast_infoset_client_policy	17-29
17.28	oracle/no_fast_infoset_service_policy	17-29
17.29	oracle/no_max_request_size_policy	17-30
17.30	oracle/no_mex_request_processing_service_policy	17-31
17.31	oracle/no_mtom_encode_fault_service_policy	17-32
17.32	oracle/no_persistence_policy	17-33
17.33	oracle/no_pox_http_binding_service_policy	17-34
17.34	oracle/no_request_processing_service_policy	17-34
17.35	oracle/no_schema_validation_policy	17-35
17.36	oracle/no_soap_request_processing_service_policy	17-36
17.37	oracle/no_test_page_processing_service_policy	17-37
17.38	oracle/no_ws_logging_level_policy	17-38
17.39	oracle/no_wsdl_request_processing_service_policy	17-39
17.40	oracle/persistence_policy	17-39
17.41	oracle/pox_http_binding_service_policy	17-40
17.42	oracle/request_processing_service_policy	17-41
17.43	oracle/schema_validation_policy	17-42
17.44	oracle/soap_request_processing_service_policy	17-43
17.45	oracle/test_page_processing_policy	17-43
17.46	oracle/ws_logging_level_policy	17-44
17.47	oracle/wsdl_request_processing_service_policy	17-45
17.48	oracle/log_policy	17-46
17.49	oracle/no_mtom_policy	17-47
17.50	oracle/wsmtom_policy	17-48
17.51	oracle/no_reliable_messaging_policy	17-49
17.52	oracle/no_wsrm_policy	17-50
17.53	oracle/reliable_messaging_policy	17-51
17.54	oracle/wsrm10_policy	17-57
17.55	oracle/wsrm11_policy	17-60
17.56	oracle/http_basic_auth_over_ssl_client_policy	17-61
17.57	oracle/http_basic_auth_over_ssl_service_policy	17-61
17.58	oracle/http_mutual_auth_over_ssl_client_policy	17-62
17.59	oracle/http_mutual_auth_over_ssl_service_policy	17-63
17.60	oracle/http_oam_token_service_policy	17-64
17.61	oracle/http_saml20_token_bearer_client_policy	17-65
17.62	oracle/http_saml20_token_bearer_service_policy	17-66

17.63	oracle/http_saml20_token_bearer_over_ssl_client_policy	17-66
17.64	oracle/http_saml20_bearer_token_over_ssl_service_policy	17-67
17.65	oracle/multi_token_rest_service_policy	17-68
17.66	oracle/multi_token_over_ssl_rest_service_policy	17-69
17.67	oracle/multi_token_sso_over_ssl_rest_service_policy	17-71
17.68	oracle/multi_token_sso_rest_service_policy	17-72
17.69	oracle/no_authentication_client_policy	17-72
17.70	oracle/no_authentication_service_policy	17-73
17.71	oracle/wss_http_token_client_policy	17-74
17.72	oracle/wss_http_token_service_policy	17-75
17.73	oracle/wss_username_token_client_policy	17-76
17.74	oracle/wss_username_token_service_policy	17-77
17.75	oracle/wss10_saml_token_client_policy	17-78
17.76	oracle/wss10_saml_token_service_policy	17-79
17.77	oracle/wss10_saml20_token_client_policy	17-80
17.78	oracle/wss10_saml20_token_service_policy	17-81
17.79	oracle/wss11_kerberos_token_client_policy	17-82
17.80	oracle/wss11_kerberos_token_service_policy	17-83
17.81	oracle/http_oauth2_token_client_policy	17-83
17.82	oracle/http_oauth2_token_with_resource_owner_creds_client_policy	17-87
17.83	oracle/http_oauth2_token_with_resource_owner_creds_over_ssl_client_policy	17-91
17.84	oracle/http_jwt_token_service_policy	17-91
17.85	oracle/http_oauth2_token_identity_switch_over_ssl_client_policy	17-93
17.86	oracle/http_jwt_token_over_ssl_service_policy	17-96
17.87	oracle/http_oauth2_token_opc_oauth2_client_policy	17-98
17.88	oracle/http_oauth2_token_over_ssl_client_policy	17-101
17.89	oracle/http_jwt_token_over_ssl_service_policy	17-105
17.90	oracle/oauth2_config_client_policy	17-105
17.91	oracle/http_jwt_token_client_policy	17-106
17.92	oracle/http_jwt_token_over_ssl_client_policy	17-108
17.93	oracle/http_oauth2_token_identity_switch_opc_oauth2_over_ssl_client_policy	17-108
17.94	oracle/http_oauth2_token_opc_oauth2_over_ssl_client_policy	17-112
17.95	oracle/http_jwt_token_identity_switch_client_policy	17-116
17.96	oracle/binding_authorization_denyall_policy	17-117
17.97	oracle/binding_authorization_permitall_policy	17-117
17.98	oracle/binding_permission_authorization_policy	17-118
17.99	oracle/component_authorization_denyall_policy	17-119
17.100	oracle/component_authorization_permitall_policy	17-120
17.101	oracle/component_permission_authorization_policy	17-120
17.102	oracle/no_authorization_component_policy	17-121
17.103	oracle/no_authorization_service_policy	17-122
17.104	oracle/whitelist_authorization_policy	17-123

17.105	oracle/no_messageprotection_client_policy	17-124
17.106	oracle/no_messageprotection_service_policy	17-125
17.107	oracle/wss10_message_protection_client_policy	17-126
17.108	oracle/wss10_message_protection_service_policy	17-128
17.109	oracle/wss11_message_protection_client_policy	17-129
17.110	oracle/wss11_message_protection_service_policy	17-130
17.111	wss11_username_token_derivedkey_with_message_protection_service	17-131
17.112	oracle/wss11_username_token_with_message_protection_client_policy	17-133
17.113	wss11_username_token_derivedkey_message_protection_encryption_client	17-134
17.114	oracle/pii_security_policy	17-136
17.115	oracle/sts_trust_config_client_policy	17-136
17.116	oracle/sts_trust_config_service_policy	17-138
17.117	oracle/wss_saml_bearer_or_username_token_service_policy	17-138
17.118	oracle/wss_saml_or_username_token_service_policy	17-139
17.119	oracle/wss_saml_or_username_token_over_ssl_service_policy	17-140
17.120	oracle/wss_saml_token_bearer_client_policy	17-140
17.121	oracle/wss_saml_token_bearer_over_ssl_client_policy	17-141
17.122	oracle/wss_saml_token_bearer_over_ssl_service_policy	17-142
17.123	oracle/wss_http_token_over_ssl_client_policy	17-143
17.124	oracle/wss_http_token_over_ssl_service_policy	17-144
17.125	oracle/wss_saml_token_over_ssl_client_policy	17-145
17.126	oracle/wss_saml_token_over_ssl_service_policy	17-146
17.127	oracle/wss_saml20_token_bearer_over_ssl_client_policy	17-147
17.128	oracle/wss_saml20_token_bearer_over_ssl_service_policy	17-147
17.129	oracle/wss_saml20_token_over_ssl_client_policy	17-148
17.130	oracle/wss_saml20_token_over_ssl_service_policy	17-149
17.131	oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy	17-150
17.132	oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy	17-151
17.133	oracle/wss_username_token_over_ssl_client_policy	17-151
17.134	oracle/wss_username_token_over_ssl_service_policy	17-152
17.135	oracle/wss_username_token_over_ssl_wssc_client_policy	17-153
17.136	oracle/wss_username_token_over_ssl_wssc_service_policy	17-154
17.137	oracle/wss_username_token_over_ssl_notimestamp_client_policy	17-155
17.138	oracle/wss_username_token_over_ssl_notimestamp_service_policy	17-156
17.139	oracle/wss10_saml_hok_token_with_message_protection_client_policy	17-157
17.140	oracle/wss10_saml_hok_token_with_message_protection_service_policy	17-158
17.141	oracle/wss10_saml_token_with_message_integrity_client_policy	17-160
17.142	oracle/wss10_saml_token_with_message_integrity_service_policy	17-161
17.143	oracle/wss10_saml_token_with_message_protection_client_policy	17-162
17.144	oracle/wss10_saml_token_with_message_protection_service_policy	17-164
17.145	oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy	17-165
17.146	oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy	17-167

17.147	oracle/wss10_saml20_token_with_message_protection_client_policy	17-169
17.148	oracle/wss10_saml20_token_with_message_protection_service_policy	17-170
17.149	oracle/wss10_username_id_propagation_with_msg_protection_client_policy	17-172
17.150	oracle/wss10_username_id_propagation_with_msg_protection_service_policy	17-173
17.151	oracle/wss10_username_token_with_message_protection_client_policy	17-174
17.152	oracle/wss10_username_token_with_message_protection_service_policy	17-176
17.153	oracle/ wss10_username_token_with_message_protection_ski_basic256_client_policy	17-177
17.154	oracle/ wss10_username_token_with_message_protection_ski_basic256_service_policy	17-179
17.155	oracle/wss10_x509_token_with_message_protection_client_policy	17-181
17.156	oracle/wss10_x509_token_with_message_protection_service_policy	17-182
17.157	oracle/wss11_kerberos_token_with_message_protection_client_policy	17-183
17.158	oracle/wss11_kerberos_token_with_message_protection_service_policy	17-184
17.159	oracle/wss11_kerberos_token_with_message_protection_basic128_client_policy	17-185
17.160	oracle/wss11_kerberos_token_with_message_protection_basic128_service_policy	17-186
17.161	oracle/wss11_saml_or_username_token_with_message_protection_service_policy	17-187
17.162	oracle/ wss11_saml_or_username_token_with_message_protection_sha256_service_policy	17-188
17.163	oracle/wss11_saml_token_identity_switch_with_message_protection_client_policy	17-190
17.164	oracle/ wss11_saml_token_identity_switch_with_message_protection_sha256_client_policy	17-191
17.165	oracle/wss11_saml_token_with_message_protection_client_policy	17-193
17.166	oracle/wss11_saml_token_with_message_protection_service_policy	17-195
17.167	oracle/wss11_saml_token_with_message_protection_sha256_client_policy	17-196
17.168	oracle/wss11_saml_token_with_message_protection_sha256_service_policy	17-197
17.169	oracle/wss11_saml_token_with_message_protection_wssc_client_policy	17-198
17.170	oracle/wss11_saml_token_with_message_protection_wssc_service_policy	17-200
17.171	oracle/wss11_saml_token_with_message_protection_wssc_reauthn_client_policy	17-201
17.172	oracle/wss11_saml_token_with_message_protection_wssc_reauthn_service_policy	17-203
17.173	oracle/wss11_saml20_token_with_message_protection_client_policy	17-204
17.174	oracle/wss11_saml20_token_with_message_protection_service_policy	17-205
17.175	oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy	17-207
17.176	oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy	17-208
17.177	oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy	17-209
17.178	oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy	17-210
17.179	oracle/wss11_sts_issued_saml_with_message_protection_client_policy	17-211
17.180	oracle/wss11_username_token_with_message_protection_client_policy	17-212
17.181	oracle/wss11_username_token_with_message_protection_service_policy	17-213
17.182	oracle/wss11_username_token_with_message_protection_sha256_client_policy	17-214
17.183	oracle/wss11_username_token_with_message_protection_sha256_service_policy	17-216
17.184	oracle/wss11_username_token_with_message_protection_wssc_client_policy	17-217
17.185	oracle/wss11_username_token_with_message_protection_wssc_service_policy	17-219

17.186	oracle/wss11_x509_token_with_message_protection_client_policy	17-220
17.187	oracle/wss11_x509_token_with_message_protection_service_policy	17-221
17.188	oracle/wss11_x509_token_with_message_protection_wssc_client_policy	17-222
17.189	oracle/wss11_x509_token_with_message_protection_wssc_service_policy	17-223
17.190	oracle/wss_saml_bearer_or_username_token_sha256_service_policy	17-224
17.191	oracle/wss_saml_token_bearer_identity_switch_client_policy	17-225
17.192	oracle/wss_saml_token_bearer_identity_switch_sha256_client_policy	17-226
17.193	oracle/wss_saml_token_bearer_over_ssl_sha256_client_policy	17-227
17.194	oracle/wss_saml_token_bearer_over_ssl_sha256_service_policy	17-228
17.195	oracle/wss_saml_token_bearer_service_policy	17-229
17.196	oracle/wss_saml_token_bearer_sha256_client_policy	17-229
17.197	oracle/wss_saml_token_bearer_sha256_service_policy	17-230
17.198	oracle/binding_oes_authorization_policy	17-230
17.199	oracle/binding_oes_masking_policy	17-231
17.200	oracle/component_oes_authorization_policy	17-231
17.201	oracle/jms_transport_client_policy	17-232
17.202	oracle/jms_transport_service_policy	17-234
17.203	oracle/no_jms_transport_client_policy	17-237
17.204	oracle/no_jms_transport_service_policy	17-238
17.205	oracle/http_oauth2_token_over_ssl_salesforce_jwt_client_policy	17-238
17.206	oracle/multi_token_rest_access_service_policy	17-240
17.207	oracle/multi_token_rest_access_over_ssl_service_policy	17-241
17.208	oracle/http_anonymous_rest_service_policy	17-242
17.209	oracle/http_anonymous_rest_over_ssl_service_policy	17-243
17.210	oracle/http_oauth2_token_over_ssl_google_jwt_client_policy	17-244
17.211	oracle/wss_saml20_token_bearer_over_ssl_notimestamp_client_policy	17-246
17.212	oracle/wss_saml20_token_bearer_over_ssl_notimestamp_service_policy	17-247
17.213	oracle/http_oauth2_token_idcs_client_policy	17-248
17.214	oracle/http_oauth2_token_over_ssl_idcs_client_policy	17-251
17.215	oracle/http_oauth2_token_identity_switch_over_ssl_idcs_client_policy	17-255

18 Oracle Web Services Manager Predefined Assertion Templates

18.1	Authentication Only Assertion Templates	18-1
18.2	Message-Protection Only Assertion Templates	18-3
18.3	Message Protection and Authentication Assertion Templates	18-3
18.4	Oracle Entitlements Server (OES) Integration Templates	18-5
18.5	PII Assertion Templates	18-5
18.6	WS-Trust Assertion Templates	18-5
18.7	Authorization Assertion Templates	18-6
18.8	Management Assertion Templates	18-6
18.9	oracle/http_oam_token_service_template	18-6

18.10	oracle/http_saml20_token_bearer_client_template	18-7
18.11	oracle/http_saml20_token_bearer_service_template	18-8
18.12	oracle/http_spnego_token_client_template	18-9
18.13	oracle/http_spnego_token_service_template	18-10
18.14	oracle/wss_http_token_client_template	18-10
18.15	oracle/wss_http_token_service_template	18-11
18.16	oracle/wss_username_token_client_template	18-12
18.17	oracle/wss_username_token_service_template	18-13
18.18	oracle/wss10_saml_token_client_template	18-14
18.19	oracle/wss10_saml_token_service_template	18-15
18.20	oracle/wss10_saml20_token_client_template	18-16
18.21	oracle/wss10_saml20_token_service_template	18-17
18.22	oracle/wss11_kerberos_token_client_template	18-17
18.23	oracle/wss11_kerberos_token_service_template	18-18
18.24	oracle/http_oauth2_token_client_template	18-19
18.25	oracle/http_jwt_token_service_template	18-26
18.26	oracle/http_oauth2_token_over_ssl_client_template	18-27
18.27	oracle/http_mutual_auth_over_ssl_client_template	18-29
18.28	oracle/http_mutual_auth_over_ssl_service_template	18-30
18.29	oracle/http_jwt_token_over_ssl_service_template	18-31
18.30	oracle/oauth2_config_client_template	18-33
18.31	oracle/http_jwt_token_client_template	18-34
18.32	oracle/http_jwt_token_over_ssl_client_template	18-38
18.33	oracle/wss10_message_protection_client_template	18-42
18.34	oracle/wss10_message_protection_service_template	18-44
18.35	oracle/wss11_message_protection_client_template	18-45
18.36	oracle/wss11_message_protection_service_template	18-46
18.37	wss11_username_token_derivedkey_message_protection_signature_client	18-47
18.38	wss11_username_token_derivedkey_message_protection_encryption_client_template	18-48
18.39	oracle/wss_http_token_over_ssl_client_template	18-50
18.40	oracle/wss_http_token_over_ssl_service_template	18-51
18.41	oracle/wss_saml_token_bearer_client_template	18-51
18.42	oracle/wss_saml_token_bearer_service_template	18-52
18.43	oracle/wss_saml_token_bearer_over_ssl_client_template	18-53
18.44	oracle/wss_saml_token_bearer_over_ssl_service_template	18-55
18.45	oracle/wss_saml20_token_bearer_over_ssl_client_template	18-56
18.46	oracle/wss_saml20_token_bearer_over_ssl_service_template	18-57
18.47	oracle/wss_saml_token_over_ssl_client_template	18-58
18.48	oracle/wss_saml_token_over_ssl_service_template	18-60
18.49	oracle/wss_saml20_token_over_ssl_client_template	18-60
18.50	oracle/wss_saml20_token_over_ssl_service_template	18-62
18.51	oracle/wss_username_token_over_ssl_client_template	18-62

18.52	oracle/wss_username_token_over_ssl_service_template	18-64
18.53	oracle/wss10_saml_hok_token_with_message_protection_client_template	18-65
18.54	oracle/wss10_saml_hok_token_with_message_protection_service_template	18-66
18.55	oracle/wss10_saml_token_with_message_protection_client_template	18-67
18.56	oracle/wss10_saml_token_with_message_protection_service_template	18-69
18.57	oracle/wss10_saml20_token_with_message_protection_client_template	18-70
18.58	oracle/wss10_saml20_token_with_message_protection_service_template	18-72
18.59	oracle/wss10_username_token_with_message_protection_client_template	18-73
18.60	oracle/wss10_username_token_with_message_protection_service_template	18-75
18.61	oracle/wss10_x509_token_with_message_protection_client_template	18-76
18.62	oracle/wss10_x509_token_with_message_protection_service_template	18-78
18.63	oracle/wss11_kerberos_token_over_ssl_client_template	18-78
18.64	oracle/wss11_kerberos_token_over_ssl_service_template	18-79
18.65	oracle/wss11_kerberos_token_with_message_protection_client_template	18-80
18.66	oracle/wss11_kerberos_token_with_message_protection_service_template	18-82
18.67	oracle/wss11_saml_token_with_message_protection_client_template	18-82
18.68	oracle/wss11_saml_token_with_message_protection_service_template	18-84
18.69	oracle/wss11_saml20_token_with_message_protection_client_template	18-85
18.70	oracle/wss11_saml20_token_with_message_protection_service_template	18-87
18.71	oracle/wss11_username_token_with_message_protection_client_template	18-88
18.72	oracle/wss11_username_token_with_message_protection_service_template	18-90
18.73	oracle/wss11_x509_token_with_message_protection_client_template	18-91
18.74	oracle/wss11_x509_token_with_message_protection_service_template	18-92
18.75	oracle/binding_oes_authorization_template	18-93
18.76	oracle/binding_oes_masking_template	18-94
18.77	oracle/component_oes_authorization_template	18-95
18.78	oracle/pii_security_template	18-95
18.79	oracle/sts_trust_config_client_template	18-96
18.80	oracle/sts_trust_config_service_template	18-97
18.81	oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template	18-98
18.82	oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template	18-100
18.83	oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template	18-101
18.84	oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template	18-103
18.85	oracle/wss11_sts_issued_saml_with_message_protection_client_template	18-104
18.86	oracle/binding_authorization_template	18-105
18.87	oracle/binding_permission_authorization_template	18-106
18.88	oracle/component_authorization_template	18-107
18.89	oracle/component_permission_authorization_template	18-108
18.90	Supported Algorithm Suites	18-109
18.91	Message Signing and Encryption Settings for Request, Response, and Fault Messages	18-112

Part VI Security and Policy Reference for Oracle Web Services

Part VII Web Server Manager Support for Web Logic Server Secure Mode

19 SSL Configuration Requirements for OWSM

19.1	WSM support for WLS Secure Mode	19-1
------	---------------------------------	------

A Security and Policy Annotations for Oracle Web Services

A.1	About Security and Policy Annotations for Web Services	A-1
A.2	Summary of Security and Policy Annotations for Web Services	A-1
A.3	List of Security and Policy Annotations for Web Services	A-3
A.3.1	@Addressing	A-4
A.3.1.1	@Addressing Attributes	A-4
A.3.1.2	@Addressing Example	A-4
A.3.2	@AtomicTransaction	A-5
A.3.2.1	@AtomicTransaction Attributes	A-5
A.3.2.2	@AtomicTransaction Example	A-6
A.3.3	@Buffering	A-6
A.3.4	@CacheBinaryContent	A-7
A.3.4.1	@CacheBinaryContent Attributes	A-7
A.3.4.2	@CacheBinaryContent Example	A-8
A.3.5	@CallbackManagementPolicy	A-8
A.3.5.1	@CallbackManagementPolicy Attributes	A-8
A.3.5.2	@CallbackManagementPolicy Example	A-9
A.3.6	@CallbackMtomPolicy	A-9
A.3.6.1	@CallbackMtomPolicy Attributes	A-9
A.3.6.2	@CallbackMtomPolicy Example	A-10
A.3.7	@CallbackPolicySet	A-10
A.3.7.1	@CallbackPolicySet Attributes	A-10
A.3.7.2	@CallbackPolicySet Example	A-10
A.3.8	@CallbackSecurityPolicy	A-11
A.3.8.1	@CallbackSecurityPolicy Attributes	A-11
A.3.8.2	@CallbackSecurityPolicy Example	A-11
A.3.9	@FastInfoSetCallbackClient	A-12
A.3.9.1	@FastInfoSetCallbackClient Attributes	A-12
A.3.9.2	@FastInfoSetCallbackClient Example	A-12
A.3.10	@FastInfoSetClient	A-13

A.3.11	@FastInfosetService	A-13
A.3.11.1	@FastInfosetService Attribute	A-13
A.3.11.2	@FastInfosetService Example	A-13
A.3.12	@JMSTransportClient	A-13
A.3.12.1	@JMSTransportClient Attributes	A-14
A.3.12.2	@JMSTransportClient Example	A-14
A.3.13	@JMSTransportService	A-14
A.3.13.1	@JMSTransportService Attributes	A-15
A.3.13.2	@JMSTransportService Example	A-15
A.3.14	@ManagementPolicy	A-15
A.3.14.1	@ManagementPolicy Attributes	A-15
A.3.14.2	@ManagementPolicy Example	A-16
A.3.15	@MaxRequestSize	A-16
A.3.15.1	@MaxRequestSize Attributes	A-16
A.3.15.2	@MaxRequestSize Example	A-16
A.3.16	@MEXRequestProcessingService	A-17
A.3.16.1	@MEXRequestProcessingService Attribute	A-17
A.3.16.2	@MEXRequestProcessingService Example	A-17
A.3.17	@MTOM	A-17
A.3.17.1	@MTOM Attribute	A-17
A.3.17.2	@MTOM Example	A-18
A.3.18	@MTOMEncodeFaultService	A-18
A.3.18.1	@MTOMEncodeFaultService Attribute	A-18
A.3.18.2	@MTOMEncodeFaultService Example	A-18
A.3.19	@MtomPolicy	A-19
A.3.19.1	@MtomPolicy Attributes	A-19
A.3.19.2	@MtomPolicy Example	A-19
A.3.20	@OAuth1 Client Policy	A-19
A.3.21	@Persistence	A-20
A.3.21.1	@Persistence Attributes	A-20
A.3.21.2	@Persistence Example	A-20
A.3.22	@PolicyReference	A-21
A.3.22.1	@PolicyReference Attributes	A-21
A.3.22.2	@PolicyReference Example	A-21
A.3.23	@PolicySet	A-22
A.3.23.1	@PolicySet Attributes	A-22
A.3.23.2	@PolicySet Example	A-22
A.3.24	@POXHttpBindingService	A-22
A.3.24.1	@POXHttpBindingService Attribute	A-22
A.3.24.2	@POXHttpBindingService Example	A-23
A.3.25	@Property	A-23
A.3.25.1	@Property Attributes	A-23

A.3.25.2	@Property Example	A-23
A.3.26	@ReliabilityPolicy	A-24
A.3.26.1	@ReliabilityPolicy Attributes	A-24
A.3.26.2	@ReliabilityPolicy Example	A-24
A.3.27	@ReliableMessaging	A-24
A.3.27.1	@ReliableMessaging Attributes	A-25
A.3.27.2	@ReliableMessaging Example	A-25
A.3.28	@RequestProcessingService	A-25
A.3.28.1	@RequestProcessingService Attribute	A-25
A.3.28.2	@RequestProcessingService Example	A-25
A.3.29	@SchemaValidation	A-26
A.3.29.1	@SchemaValidation Attribute	A-26
A.3.29.2	@SchemaValidation Example	A-26
A.3.30	@SecurityPolicies (Oracle Infrastructure Web Services)	A-26
A.3.31	@SecurityPolicies (Java EE Web Services)	A-27
A.3.31.1	@SecurityPolicies Example	A-27
A.3.32	@SecurityPolicy (Oracle Infrastructure Web Services)	A-27
A.3.32.1	@SecurityPolicy Attributes	A-27
A.3.32.2	@SecurityPolicy Example	A-28
A.3.33	@SecurityPolicy (Java EE Web Services)	A-28
A.3.33.1	@SecurityPolicy Attributes	A-28
A.3.33.2	@SecurityPolicy Example	A-28
A.3.34	@SOAPRequestProcessingService	A-28
A.3.34.1	@SOAPRequestProcessingService Attribute	A-29
A.3.34.2	@SOAPRequestProcessingService Example	A-29
A.3.35	@TestPageProcessingService	A-29
A.3.35.1	@TestPageProcessingService Attribute	A-29
A.3.35.2	@TestPageProcessingService Example	A-29
A.3.36	@WSDLRequestProcessingService	A-30
A.3.36.1	@WSDLRequestProcessingService Attribute	A-30
A.3.36.2	@WSDLRequestProcessingService Example	A-30
A.3.37	@WSLoggingLevel	A-30
A.3.37.1	@WSLoggingLevel Attributes	A-30
A.3.37.2	@WSLoggingLevel Example	A-31

B Predefined Assertion Templates for Oracle Web Services

B.1	Assertion Template Settings for Oracle Web Services	B-1
B.1.1	Action Match	B-3
B.1.2	Algorithm Suite	B-3
B.1.3	Authentication Header—Header Name	B-3
B.1.4	Authentication Header—Mechanism	B-3

B.1.5	Body Elements	B-4
B.1.6	Bootstrap Message Security	B-4
B.1.7	Client Entropy	B-4
B.1.8	Client Policy URI	B-4
B.1.9	Confirm Signature	B-5
B.1.10	Confirmation Type	B-5
B.1.11	Constraint Match	B-5
B.1.12	Creation Time Required	B-5
B.1.13	Derived Keys	B-5
B.1.14	Enabled	B-5
B.1.15	Encrypt Signature	B-5
B.1.16	Encryption Key Reference Mechanism	B-6
B.1.17	Fault	B-6
B.1.18	Fault Message Settings	B-6
B.1.19	Header Elements	B-7
B.1.20	Include Entire Body	B-7
B.1.21	Include MIME Headers	B-7
B.1.22	Include SwA Attachment	B-7
B.1.23	Include Timestamp	B-7
B.1.24	Is Encrypted	B-8
B.1.25	Is Signed	B-8
B.1.26	Kerberos Token Type	B-8
B.1.27	Key Type	B-8
B.1.28	Keystore Recipient Alias	B-8
B.1.29	Mutual Authentication Required	B-8
B.1.30	Name Identifier Format	B-8
B.1.31	Nonce Required	B-9
B.1.32	Password Type	B-9
B.1.33	Permissions	B-9
B.1.34	Permission Class	B-10
B.1.35	Port Endpoint	B-10
B.1.36	Port URI	B-10
B.1.37	Re-authenticate	B-10
B.1.38	Recipient Encryption Key Reference Mechanism	B-10
B.1.39	Recipient Sign Key Reference Mechanism	B-10
B.1.40	Request	B-11
B.1.41	Request Message Settings	B-11
B.1.42	Request XPath	B-11
B.1.43	Request Namespaces	B-11
B.1.44	Require Applies To	B-11
B.1.45	Require Client Entropy	B-11
B.1.46	Require External Reference	B-11

B.1.47	Require Internal Reference	B-11
B.1.48	Require Server Entropy	B-12
B.1.49	Resource Match	B-12
B.1.50	Response	B-12
B.1.51	Response Message Settings	B-12
B.1.52	Response Namespaces	B-12
B.1.53	Response XPath	B-12
B.1.54	Roles	B-12
B.1.55	Server Entropy	B-13
B.1.56	Sign Key Reference Mechanism	B-13
B.1.57	Sign Then Encrypt	B-13
B.1.58	Token Type	B-13
B.1.59	Transport Layer Security	B-13
B.1.60	Transport Layer Security—Include Timestamp	B-13
B.1.61	Transport Layer Security—Mutual Authentication Required	B-14
B.1.62	Version	B-14
B.1.63	Trust Version	B-14
B.1.64	Use Derived Keys	B-14
B.1.65	Use PKI Path	B-14
B.1.66	WSDL Exist	B-14
B.1.67	WSDL	B-14
B.2	Assertion Template Configuration Properties for Oracle Web Services	B-15
B.2.1	algorithm	B-16
B.2.2	anonymous.access	B-16
B.2.3	application.name	B-17
B.2.4	attesting.mapping.attribute	B-17
B.2.5	auth.header.token.type	B-17
B.2.6	caller.principal.name	B-17
B.2.7	credential.delegation	B-17
B.2.8	csf.map	B-17
B.2.9	csf-key	B-17
B.2.10	encryption-algorithm	B-18
B.2.11	execute.action	B-18
B.2.12	ignore.timestamp.in.response	B-18
B.2.13	include-timestamp	B-18
B.2.14	issued.token.caching	B-18
B.2.15	issued.token.lifetime	B-18
B.2.16	iteration	B-18
B.2.17	iterations	B-19
B.2.18	keysize	B-19
B.2.19	keytab.location	B-19
B.2.20	keystore.enc.csf.key	B-19

B.2.21	keystore.recipient.alias	B-19
B.2.22	keystore.sig.csf.key	B-19
B.2.23	lookup.action	B-19
B.2.24	on.behalf.of	B-19
B.2.25	policy.reference.uri	B-20
B.2.26	port.endpoint	B-20
B.2.27	port.uri	B-20
B.2.28	propagate.identity.context	B-20
B.2.29	realm	B-20
B.2.30	reference.priority	B-20
B.2.31	remote-user	B-21
B.2.32	resource.mapping.model	B-21
B.2.33	resource.name	B-21
B.2.34	resource.type	B-21
B.2.35	rm.encrypt.body	B-21
B.2.36	role	B-21
B.2.37	salt	B-21
B.2.38	saml.assertion.filename	B-22
B.2.39	saml.audience.uri	B-22
B.2.40	saml.envelope.signature.required	B-22
B.2.41	saml.issuer.name	B-22
B.2.42	saml.trusted.issuers	B-22
B.2.43	sc.token.lifetime	B-22
B.2.44	service.principal.name	B-22
B.2.45	subject.precedence	B-22
B.2.46	sts.auth.caller.principal.name	B-23
B.2.47	sts.auth.keytab.location	B-23
B.2.48	sts.auth.on.behalf.of.csf.key	B-23
B.2.49	sts.auth.on.behalf.of.username.only	B-23
B.2.50	sts.auth.service.principal.name	B-23
B.2.51	sts.auth.user.csf.key	B-23
B.2.52	sts.auth.x509.csf.key	B-23
B.2.53	sts.in.order	B-23
B.2.54	sts.keystore.recipient.alias	B-24
B.2.55	user.csf.key	B-24
B.2.56	use.single.step	B-24
B.2.57	user.attributes	B-24
B.2.58	user.roles.include	B-25
B.2.59	user.tenant.name	B-25
B.2.60	wSDL.uri	B-25

C Schema Reference for Predefined Assertions for Oracle Web Services

C.1	wsp:Policy Element	C-4
C.1.1	WS-Policy Attributes	C-4
C.1.2	Example of WS-Policy	C-5
C.2	wsp:ExactlyOne Element	C-6
C.2.1	wsp:ExactlyOne Element Attribute	C-6
C.2.2	Example of wsp:ExactlyOne Element	C-6
C.3	orasp:Assertion Element	C-7
C.3.1	orasp:Assertion Element Attributes	C-8
C.3.2	Example of orasp:Assertion Element	C-8
C.4	orawsp:bindings Element	C-8
C.4.1	Example of orawsp:bindings Element	C-8
C.5	orawsp:Config Element	C-9
C.5.1	orawsp:Config Element Attributes	C-9
C.5.2	Example of orawsp:Config Element	C-9
C.6	orawsp:PropertySet Element	C-9
C.6.1	orawsp:PropertySet Element Attributes	C-9
C.6.2	Example of orawsp:PropertySet Element	C-10
C.7	orawsp:Property Element	C-10
C.7.1	orawsp:Property Element Attributes	C-10
C.7.2	Example of orawsp:Property Element	C-13
C.8	orawsp:Description Element	C-13
C.8.1	Example of orawsp:Description Element	C-13
C.9	orawsp:Value Element	C-13
C.9.1	Example of orawsp:Value Element	C-13
C.10	orawsp:guard Element	C-13
C.10.1	Examples of orawsp:guard Element	C-14
C.11	orawsp:resource-match Element	C-14
C.11.1	Examples of orawsp:resource-match	C-14
C.12	orawsp:action-match Element	C-14
C.12.1	Examples of orawsp:action-match Element	C-14
C.13	orawsp:constraint-match Element	C-15
C.13.1	Example of orawsp:constraint-match Element	C-15
C.14	oralgp:Logging Element	C-15
C.14.1	Example of oralgp:Logging Element	C-15
C.15	orasp:binding-authorization Element	C-16
C.15.1	Example of orasp:binding-authorization Element	C-16
C.16	orasp:binding-permission-authorization Element	C-16
C.16.1	Example of orasp:binding-permission-authorization Element	C-16
C.17	orasp:coreid-security Element	C-17
C.17.1	Example of orasp:coreid-security Element	C-17

C.18	orasp:http-security Element	C-17
C.18.1	Example of orasp:http-security Element	C-18
C.19	orasp:kerberos-security Element	C-18
C.19.1	Example of orasp:kerberos-security Element	C-18
C.20	orasp:sca-component-authorization Element	C-18
C.20.1	Example of orasp:sca-component-authorization Element	C-19
C.21	orasp:sca-component-permission-authorization Element	C-19
C.21.1	Example of orasp:sca-component-permission-authorization Element	C-19
C.22	orasp:sts-trust-config Element	C-20
C.22.1	orasp:sts-trust-config Element Attributes	C-20
C.22.2	Example of orasp:sts-trust-config Element	C-20
C.23	orasp:wss10-anonymous-with-certificates Element	C-21
C.23.1	Example of orasp:wss10-anonymous-with-certificates Element	C-21
C.24	orasp:wss10-mutual-auth-with-certificates Element	C-22
C.24.1	Example of orasp:wss10-mutual-auth-with-certificates Element	C-22
C.25	orasp:wss10-saml-hok-with-certificates Element	C-23
C.25.1	Example of orasp:wss10-saml-hok-with-certificates Element	C-23
C.26	orasp:wss10-saml-token Element	C-24
C.26.1	Example of orasp:wss10-saml-token Element	C-24
C.27	orasp:wss10-saml-with-certificates Element	C-24
C.27.1	Example of orasp:wss10-saml-with-certificates Element	C-25
C.28	orasp:wss10-username-with-certificates Element	C-25
C.28.1	Example of orasp:wss10-username-with-certificates Element	C-26
C.29	orasp:wss11-anonymous-with-certificates Element	C-26
C.29.1	Example of orasp:wss11-anonymous-with-certificates Element	C-27
C.30	orasp:wss11-mutual-auth-with-certificates Element	C-27
C.30.1	Example of orasp:wss11-mutual-auth-with-certificates Element	C-28
C.31	orasp:wss11-saml-with-certificates Element	C-28
C.31.1	Example of orasp:wss11-saml-with-certificates Element	C-29
C.32	orasp:wss11-sts-issued-token-with-certificates Element	C-29
C.32.1	orasp:wss11-sts-issued-token-with-certificates Element Attributes	C-30
C.32.2	Example of orasp:wss11-sts-issued-token-with-certificates Element	C-30
C.33	orasp:wss11-username-with-certificates Element	C-31
C.33.1	Example of orasp:wss11-username-with-certificates Element	C-32
C.34	orasp:wss-saml-token-bearer-over-ssl Element	C-32
C.34.1	Example of orasp:wss-saml-token-bearer-over-ssl Element	C-33
C.35	orasp:wss-saml-token-over-ssl Element	C-33
C.35.1	Example of orasp:wss-saml-token-over-ssl Element	C-33
C.36	orasp:wss-sts-issued-token-over-ssl Element	C-34
C.36.1	orasp:wss-sts-issued-token-over-ssl Element Attributes	C-34
C.36.2	Example of orasp:wss-sts-issued-token-over-ssl Element	C-34
C.37	orasp:wss-username-token Element	C-35

C.37.1	Example of orasp:wss-username-token Element	C-35
C.38	orasp:wss-username-token-over-ssl Element	C-35
C.38.1	Example of orasp:wss-username-token-over-ssl Element	C-36
C.39	rm:RMAssertion Element	C-36
C.39.1	Example of rm:RMAssertion Element	C-36
C.40	wsaw:UsingAddressing Element	C-37
C.40.1	Example of wsaw:UsingAddressing Element	C-37
C.41	wsoma:OptimizedMimeSerialization Element	C-38
C.41.1	Example of wsoma:OptimizedMimeSerialization Element	C-38
C.42	oralgp:fault Element	C-38
C.42.1	Example of oralgp:fault Element	C-38
C.43	oralgp:request Element	C-38
C.43.1	Example of oralgp:request Element	C-39
C.44	oralgp:response Element	C-39
C.44.1	Example of oralgp:response Element	C-39
C.45	oralgp:msg-log Element	C-39
C.45.1	Example of oralgp:msg-log Element	C-39
C.46	orasp:attachment Element	C-39
C.46.1	orasp:attachment Element Attributes	C-39
C.46.2	Example of orasp:attachment Element	C-40
C.47	orasp:auth-header Element	C-40
C.47.1	orasp:auth-header Element Attributes	C-40
C.47.2	Example of rasp:auth-header Element	C-40
C.48	orasp:body Element	C-40
C.48.1	Example of orasp:body Element	C-40
C.49	orasp:check-permission Element	C-41
C.49.1	Example of orasp:check-permission Element	C-41
C.50	orasp:coreid-token Element	C-41
C.50.1	orasp:coreid-token Element Attributes	C-41
C.50.2	Example of orasp:coreid-token Element	C-41
C.51	orasp:denyAll Element	C-41
C.51.1	Example of orasp:denyAll Element	C-41
C.52	orasp:element Element	C-42
C.52.1	orasp:element Element Attributes	C-42
C.52.2	Example of orasp:element Element	C-42
C.53	orasp:encrypted-elements Element	C-42
C.53.1	Example of orasp:encrypted-elements Element	C-42
C.54	orasp:encrypted-parts Element	C-42
C.54.1	Example of orasp:encrypted-parts Element	C-43
C.55	orasp:fault Element	C-43
C.55.1	Example of orasp:fault Element	C-43
C.56	orasp:header Element	C-43

C.56.1	orasp:header Element Attributes	C-43
C.56.2	Example of orasp:header Element	C-43
C.57	orasp:issued-token Element	C-44
C.57.1	orasp:issued-token Element Attributes	C-44
C.57.2	Example of orasp:issued-token Element	C-44
C.58	orasp:kerberos-token Element	C-44
C.58.1	orasp:kerberos-token Element Attributes	C-44
C.58.2	Example of orasp:kerberos-token Element	C-44
C.59	orasp:msg-security Element	C-45
C.59.1	orasp:msg-security Element Attributes	C-45
C.59.2	Example of orasp:msg-security Element	C-45
C.60	orasp:permitAll Element	C-46
C.60.1	Example of orasp:permitAll Element	C-46
C.61	orasp:request Element	C-46
C.61.1	Example of orasp:request Element	C-46
C.62	orasp:require-tls Element	C-46
C.62.1	orasp:require-tls Element Attributes	C-46
C.62.2	Example of orasp:require-tls Element	C-47
C.63	orasp:response Element	C-47
C.63.1	Example of orasp:response Element	C-47
C.64	orasp:role Element	C-47
C.64.1	orasp:role Element Attribute	C-47
C.64.2	Example of orasp:role Element	C-48
C.65	orasp:saml-token Element	C-48
C.65.1	orasp:saml-token Element Attributes	C-48
C.65.2	Example of orasp:saml-token Element	C-49
C.66	orasp:signed-elements Element	C-49
C.66.1	Example of orasp:signed-elements Element	C-49
C.67	orasp:signed-parts Element	C-49
C.67.1	Example of orasp:signed-parts Element	C-49
C.68	orasp:username-token Element	C-49
C.68.1	orasp:username-token Element Attributes	C-49
C.68.2	Example of orasp:username-token Element	C-50
C.69	orasp:x509-token Element	C-50
C.69.1	orasp:x509-token Element Attributes	C-50
C.69.2	Example of orasp:x509-token Element	C-51
C.70	orawsp:Description Element	C-51
C.70.1	Example of orawsp:Description Element	C-51

D Schema Reference for Web Services Policy Sets

D.1	policySet Element	D-1
-----	-------------------	-----

D.2	wsp:policyReference Element	D-2
D.3	orawsp:OverrideProperty Element	D-3

E Oracle Web Services Manager Introspection Plug-in for Oracle Virtual Assembly Builder

E.1	About the OWSM Introspection Plug-in for Oracle Virtual Assembly Builder	E-1
E.2	Understanding the OWSM Introspection Plug-in	E-1
E.2.1	OWSM Introspection Plug-in Parameter	E-2
E.2.2	OWSM Introspection Plug-in Reference System Prerequisites	E-2
E.2.3	OWSM Introspection Plug-in Usage Requirements	E-2
E.2.4	OWSM Introspection Plug-in Resulting Artifact Type	E-3
E.2.5	OWSM Introspection Plug-in Wiring	E-3
E.2.6	OWSM Introspection Plug-in Wiring Properties	E-3
E.2.7	OWSM Introspection Plug-in Appliance Properties	E-3
E.2.8	OWSM Introspection Plug-in Supported Template Types	E-5

F Twitter REST APIs

F.1	Twitter REST Client sample code for POST request	F-1
F.2	Twitter REST Client sample code for GET request	F-3

List of Figures

4-1	WSM Policy Subject Configuration Page with Directly Attached Policies	4-13
4-2	Attaching Policies to a Web Service	4-14
4-3	Viewing Details about a Policy	4-15
4-4	Java EE Web Service Clients	4-19
4-5	Attaching Policies to WebLogic Java EE Web Service Client Ports	4-20
4-6	WSM Policy Set Summary Page	4-22
4-7	Viewing a Policy Set	4-23
4-8	Enter General Information Page	4-24
4-9	Add Policy References Page	4-26
4-10	Summary Page in Create Policy Set Wizard	4-26
4-11	Enter Constraint Page in Policy Set Wizard	4-29
4-12	Policies Attached to an Oracle Infrastructure Web Service Endpoint	4-31
4-13	Policies Attached to a WebLogic Java EE Web Service Endpoint	4-31
4-14	Effective Policy Calculation for Policy Sets with Run Time Constraints	4-79
4-15	Web Service Endpoint Page With Valid and Secure Endpoint Configuration	4-89
5-1	Overriding a Policy Configuration Property	5-11
5-2	Policy Set Override Policy Configuration Page	5-13
6-1	WSM Policies Page	6-2
6-2	Policy Details Page with the General Tab Selected	6-5
6-3	Policy Details Page with the Assertion Tab Selected	6-6
6-4	Generate Client Policies from WSDL	6-13
6-5	Policy Version History Page	6-20
6-6	Security Assertion with Two Subcategories	6-22
6-7	Assertion Template Details Page	6-26
6-8	Enabling Advertising for Policy Assertions	6-34
7-1	Create Stripe	7-3
7-2	Create Keystore	7-4
7-3	Manage Certificates	7-4
7-4	Generate Keypair	7-5
7-5	Credential Store Provider Configuration Page	7-12
7-6	Keys Configured in OWSM Credential Map	7-13
7-7	Create Key Dialog Box	7-14
7-8	Edit Configure Property Dialog Box	7-17
7-9	Cluster Topology with Storage-Enabled and -Disabled WebLogic Servers	7-25
7-10	Cluster Topology with Storage-Enabled Managed Coherence Servers within the Coherence Cluster	7-27
7-11	Example of Partial Encryption of Message Protection Policies	7-30

8-1	Where to Attach the pii_security_policy for a SOA Composite	8-3
8-2	SOA Composite Application Dashboard Page	8-4
8-3	Overriding request.xpath Attribute	8-5
9-1	Create Keystore	9-3
9-2	Manage Certificates	9-3
9-3	Generate Keypair	9-4
9-4	Create Keystore	9-5
9-5	Manage Certificates	9-6
9-6	Generate Keypair	9-6
10-1	The Permission Settings for a Policy	10-4
10-2	Adding a Permission on the OPSS Create Application Grant Page	10-4
10-3	Create OES Application	10-8
10-4	Adding Authorization Policy for Obligations	10-9
10-5	Sample Obligations For Authorization Policy	10-10
10-6	Create New Attributes	10-12
10-7	Adding Actual Authorization Policy	10-12
10-8	Create the Condition	10-13
10-9	Create OES Application	10-14
10-10	Add Implicit Attributes Needed for Conditions	10-15
10-11	Adding Actual Authorization Policy	10-16
10-12	Create OES Application	10-17
11-1	wsee:PasswordDigest as Active Type	11-4
11-2	Selecting Digest From the Password Type Drop-Down Menu	11-4
11-3	Identity Store Configuration Page	11-9
11-4	Adding the virtualize property	11-9
11-5	Editing the WSIdentityPermission	11-14
11-6	Setting the propagate.identity.context Property to True	11-17
13-1	Sample Algorithm Suite	13-8
14-1	SAML Trust Section of Authentication Tab	14-5
14-2	JWT Trust Section of Authentication Tab	14-6
14-3	Custom Login Module Page	14-12
14-4	KSS Keystore Section of Message Security Page	14-14
14-5	JKS Section of Message Security Key Store Page	14-15
14-6	HSM Settings for OWSM Keystore Configuration	14-17
14-7	PKCS11 Section of Message Security Key Store Page	14-18
14-8	Security settings in Message Security page	14-19
14-9	Disabling Auto Refresh Option to Manually Refresh Configuration Cache in OWSM	14-27

14-10	Policy Access Tab in WSM Domain Configuration Page Showing the Status of Auto Refresh	14-28
14-11	Policy Access Tab in WSM Domain Configuration Page	14-28
14-12	Refresh Status of Servers	14-29
15-1	Metadata Repository in Navigation Pane	15-2
15-2	Registering an OWSM Repository	15-2
16-1	OWSM Policy Manager Page	16-2
16-2	OWSM Policy Manager Shutdown (Farm Page)	16-4
C-1	Element Hierarchy of an Assertion	C-2
D-1	Element Hierarchy of the Policy Set	D-1

List of Tables

1-1	Categories of Oracle Web Services Secured Using OWSM	1-1
2-1	Default OWSM Logical Roles	2-8
3-1	Choosing the Right Authentication Policy	3-2
3-2	Authentication Only Policies—SOAP and RESTful Web Services	3-4
3-3	Authentication Only Policies—SOAP Web Services Only	3-5
3-4	Authentication Only Policies—OAuth2 and JWT Web Services	3-6
3-5	Message-Protection Only Policies	3-6
3-6	Message Protection and Authentication Policies	3-7
3-7	Authorization Only Policies	3-9
3-8	WS-Trust Policies	3-10
3-9	OWSM Security Policies Supported for RESTful Web Services and Clients	3-13
3-10	Unsupported Policies and Assertions for Oracle Service Bus	3-20
4-1	Feature Classes Used for Attaching Policies to RESTful Clients	4-8
4-2	Supported Expressions for the Resource Scope	4-81
4-3	Valid Values for local.policy.reference.source Configuration Property	4-93
5-1	Client Policy Configuration Properties That Can Be Overridden at Design Time	5-4
6-1	Policy Advertisement	6-35
10-1	OWSM OES Configuration Properties	10-20
10-2	User Credential, Subject, and Access Token	10-24
11-1	SAML Token Validation	11-7
14-1	Subject Domain Configuration Properties	14-41
14-2	SAML and SAML2 Login Module Domain Configuration Properties	14-41
14-3	Kerberos Login Module Domain Configuration Properties	14-43
14-4	X509 Login Module Domain-Level Configuration Properties	14-44
14-5	Other Login Module Domain Configuration Properties	14-44
14-6	Security Policy Enforcement Domain-Level Configuration Properties	14-47
14-7	Identity Domain-Level Configuration Properties	14-49
14-8	Secure Conversation Domain-Level Configuration Properties	14-49
14-9	Policy Manager Connection Domain-Level Configuration Properties	14-50
14-10	High Availability and Cache Management Domain Configuration Properties	14-55
16-1	Common OWSM Exceptions and Errors for WS-Trust Use Cases	16-23
17-1	Configuration Property for oracle/wsaddr_policy	17-16
17-2	Configuration Property for oracle/no_addressing_policy	17-17
17-3	Configuration Properties for oracle/atomic_transaction_policy	17-18
17-4	Configuration Property for oracle/no_atomic_transaction_policy	17-19
17-5	Configuration Property for oracle/async_web_service_policy	17-20

17-6	Configuration Properties for oracle/cache_binary_content_policy	17-22
17-7	Configuration Properties for oracle/fastinfoset_client_policy	17-23
17-8	Configuration Properties for oracle/fastinfoset_service_policy	17-24
17-9	Configuration Properties for oracle/max_request_size_policy	17-25
17-10	Configuration Properties for oracle/mex_request_processing_service_policy	17-26
17-11	Configuration Properties for oracle/mtom_encode_fault_service_policy	17-27
17-12	Configuration Property for oracle/no_async_web_service_policy	17-28
17-13	Configuration Property for oracle/no_cache_binary_content_policy	17-28
17-14	Configuration Property for oracle/no_fast_infoset_client_policy	17-29
17-15	Configuration Property for oracle/no_fast_infoset_service_policy	17-30
17-16	Configuration Property for oracle/no_max_request_size_policy	17-31
17-17	Configuration Property for oracle/no_mex_request_processing_service_policy	17-32
17-18	Configuration Property for oracle/no_mtom_encode_fault_service_policy	17-33
17-19	Configuration Property for oracle/no_persistence_policy	17-33
17-20	Configuration Property for oracle/no_pox_http_binding_service_policy	17-34
17-21	Configuration Property for oracle/no_request_processing_service_policy	17-35
17-22	Configuration Property for oracle/no_schema_validation_policy	17-36
17-23	Configuration Property for oracle/no_soap_request_processing_service_policy	17-37
17-24	Configuration Property for oracle/no_test_page_processing_service_policy	17-38
17-25	Configuration Property for oracle/no_ws_logging_level_policy	17-38
17-26	Configuration Property for oracle/no_wsdl_request_processing_service_policy	17-39
17-27	Configuration Properties for oracle/persistence_policy	17-40
17-28	Configuration Property for oracle/pox_http_binding_service_policy	17-41
17-29	Configuration Property for oracle/request_processing_service_policy	17-42
17-30	Configuration Property for oracle/schema_validation_policy	17-43
17-31	Configuration Property for oracle/soap_request_processing_service_policy	17-43
17-32	Configuration Property for oracle/test_page_processing_policy	17-44
17-33	Configuration Property for oracle/ws_logging_level_policy	17-45
17-34	Configuration Property for oracle/ws_logging_level_policy	17-46
17-35	Configuration Property for oracle/log_policy	17-47
17-36	Configuration Property for oracle/no_mtom_policy	17-47
17-37	Configuration Property for oracle/wsmtom_policy	17-49
17-38	Configuration Property for oracle/no_reliable_messaging_policy	17-49
17-39	Configuration Property for oracle/no_wsrn_policy	17-50
17-40	Configuration Properties for oracle/reliable_messaging_policy	17-52
17-41	Configuration Properties for the wsrn10_policy	17-59
17-42	Configuration Property for oracle/no_authentication_client_policy	17-73

17-43	Configuration Property for oracle/no_authentication_service_policy	17-74
17-44	Configuration Property for oracle/no_authorization_component_policy	17-122
17-45	Configuration Property for oracle/no_authorization_service_policy	17-123
17-46	Configuration Property for oracle/no_messageprotection_client_policy	17-125
17-47	Configuration Property for oracle/no_messageprotection_service_policy	17-126
17-48	Configuration Properties for oracle/jms_transport_client_policy	17-232
17-49	Configuration Properties for oracle/jms_transport_service_policy	17-235
17-50	Configuration Property for oracle/no_jms_transport_client_policy	17-237
17-51	Configuration Property for oracle/no_jms_transport_service_policy	17-238
17-52	Configuration Property for oracle/multi_token_rest_access_service_policy	17-241
17-53	Configuration Property for oracle/multi_token_rest_access_over_ssl_service_policy	17-242
17-54	Configuration Property for oracle/http_anonymous_rest_service_policy	17-243
17-55	Configuration Property for oracle/http_anonymous_rest_over_ssl_service_policy	17-244
18-1	Authentication Only Assertion Templates	18-2
18-2	Message-Protection Only Assertion Templates	18-3
18-3	Message Protection and Authentication Assertion Templates	18-3
18-4	http_oam_token_service_template Settings	18-7
18-5	http_oam_token_service_template Configuration Properties	18-7
18-6	http_saml20_token_bearer_client_template Settings	18-8
18-7	http_saml20_token_bearer_client_template Configuration Properties	18-8
18-8	http_saml20_token_bearer_service_template Configuration Properties	18-9
18-9	http_spnego_token_client_template Settings	18-9
18-10	http_spnego_token_client_template Configuration Properties	18-9
18-11	http_spnego_token_service_template Configuration Properties	18-10
18-12	wss_http_token_client_template Settings	18-11
18-13	wss_http_token_client_template Configuration Properties	18-11
18-14	wss_http_token_service_template Configuration Properties	18-12
18-15	wss_username_token_client_template Settings	18-13
18-16	wss_username_token_client_template Configuration Properties	18-13
18-17	wss_username_token_service_template Configuration Properties	18-14
18-18	wss10_saml_token_client_template Settings	18-14
18-19	wss10_saml_token_client_template Configuration Properties	18-15
18-20	wss10_saml_token_service_template Configuration Properties	18-15
18-21	wss10_saml20_token_client_template Settings	18-16
18-22	wss10_saml20_token_client_template Configuration Properties	18-16
18-23	wss10_saml20_token_service_template Configuration Properties	18-17
18-24	wss11_kerberos_token_client_template Settings	18-18

18-25	wss11_kerberos_token_client_template Configuration Properties	18-18
18-26	wss11_kerberos_token_service_template Configuration Properties	18-19
18-27	http_oauth2_token_client_template Settings	18-20
18-28	http_oauth2_token_client_template Configuration Properties	18-21
18-29	http_jwt_token_service_template Configuration Properties	18-26
18-30	http_oauth2_token_over_ssl_client_template Settings	18-28
18-31	http_mutual_auth_over_ssl_client_template Settings	18-30
18-32	wss_http_token_over_ssl_client_template Configuration Properties	18-30
18-33	http_mutual_auth_over_ssl_service_template Configuration Properties	18-31
18-34	http_jwt_token_over_ssl_service_template Configuration Properties	18-32
18-35	oauth2_config_client_template Settings	18-33
18-36	oauth2_config_client_template Configuration Properties	18-33
18-37	http_jwt_token_client_template Settings	18-35
18-38	http_jwt_token_client_template Configuration Properties	18-36
18-39	http_jwt_token_over_ssl_client_template Settings	18-39
18-40	http_jwt_token_over_ssl_client_template Configuration Properties	18-40
18-41	wss10_message_protection_client_template Settings	18-43
18-42	wss10_message_protection_client_template Configuration Properties	18-44
18-43	wss10_message_protection_service_template Configuration Properties	18-44
18-44	wss11_message_protection_client_template Settings	18-45
18-45	wss11_message_protection_client_template Configuration Properties	18-46
18-46	wss11_message_protection_service_template Configuration Properties	18-46
18-47	wss11_username_token_derivedkey_with_message_protection_signature_only_client_template	18-47
18-48	wss11_username_token_derivedkey_with_message_protection_signature_only_client_template Configuration Properties	18-48
18-49	wss11_username_token_derivedkey_with_message_protection_encryption_only_client_template	18-49
18-50	wss11_username_token_derivedkey_with_message_protection_encryption_only_client_template Configuration Properties	18-50
18-51	wss_http_token_over_ssl_client_template Settings	18-50
18-52	wss_http_token_over_ssl_client_template Configuration Properties	18-51
18-53	wss_http_token_over_ssl_service_template Configuration Properties	18-51
18-54	wss_saml_token_bearer_client_template Settings	18-52
18-55	wss_saml_token_bearer_client_template Configuration Properties	18-52
18-56	wss_saml_token_bearer_service_template Settings	18-53
18-57	wss_saml_token_bearer_service_template Configuration Properties	18-53
18-58	wss_saml_token_bearer_over_ssl_client_template Settings	18-54
18-59	wss_saml_token_bearer_over_ssl_client_template Configuration Properties	18-54

18-60	wss_saml_token_bearer_over_ssl_service_template Configuration Properties	18-55
18-61	wss_saml20_token_bearer_over_ssl_client_template Settings	18-56
18-62	wss_saml20_token_bearer_over_ssl_client_template Configuration Properties	18-57
18-63	wss_saml20_token_bearer_over_ssl_service_template Configuration Properties	18-58
18-64	wss_saml_token_over_ssl_client_template Settings	18-58
18-65	wss_saml_token_over_ssl_client_template Configuration Properties	18-59
18-66	wss_saml_token_over_ssl_service_template Configuration Properties	18-60
18-67	wss_saml20_token_over_ssl_client_template Settings	18-61
18-68	wss_saml20_token_over_ssl_client_template Configuration Properties	18-61
18-69	wss_saml20_token_over_ssl_service_template Configuration Properties	18-62
18-70	wss_username_token_over_ssl_client_template Settings	18-63
18-71	wss_username_token_over_ssl_client_template Configuration Properties	18-63
18-72	wss_username_token_over_ssl_service_template Configuration Properties	18-64
18-73	wss10_saml_hok_token_with_message_protection_client_template Settings	18-65
18-74	wss10_saml_hok_token_with_message_protection_client_template Configuration Properties	18-66
18-75	wss10_saml_hok_token_with_message_protection_service_template Configuration Properties	18-67
18-76	wss10_saml_token_with_message_protection_client_template Settings	18-68
18-77	wss10_saml_token_with_message_protection_client_template Configuration Properties	18-69
18-78	wss10_saml_token_with_message_protection_service_template Configuration Properties	18-70
18-79	wss10_saml20_token_with_message_protection_client_template Settings	18-70
18-80	wss10_saml20_token_with_message_protection_client_template Configuration Properties	18-71
18-81	wss10_saml20_token_with_message_protection_service_template Configuration Properties	18-73
18-82	wss10_username_token_with_message_protection_client_template Settings	18-74
18-83	wss10_username_token_with_message_protection_client_template Configuration Properties	18-75
18-84	wss10_username_token_with_message_protection_service_template Configuration Properties	18-76
18-85	wss10_x509_token_with_message_protection_client_template Settings	18-76
18-86	wss10_x509_token_with_message_protection_client_template Configuration Properties	18-77
18-87	wss10_x509_token_with_message_protection_service_template Configuration Properties	18-78
18-88	wss11_kerberos_token_over_ssl_client_template Settings	18-79
18-89	wss11_kerberos_token_over_ssl_client_template Configuration Properties	18-79
18-90	wss11_kerberos_token_over_ssl_service_template Configuration Properties	18-80
18-91	wss11_kerberos_token_with_message_protection_client_template Settings	18-81
18-92	wss11_kerberos_token_with_message_protection_client_template Configuration Properties	18-81
18-93	wss11_kerberos_token_with_message_protection_service_template Configuration Properties	18-82
18-94	wss11_saml_token_with_message_protection_client_template Settings	18-83
18-95	wss11_saml_token_with_message_protection_client_template Configuration Properties	18-84
18-96	wss11_saml_token_with_message_protection_service_template Configuration Properties	18-85

18-97	wss11_saml20_token_with_message_protection_client_template Settings	18-85
18-98	wss11_saml20_token_with_message_protection_client_template Configuration Properties	18-86
18-99	wss11_saml20_token_with_message_protection_service_template Configuration Properties	18-88
18-100	wss11_username_token_with_message_protection_client_template Settings	18-88
18-101	wss11_username_token_with_message_protection_client_template Configuration Properties	18-89
18-102	wss11_username_token_with_message_protection_service_template Configuration Properties	18-90
18-103	wss11_x509_token_with_message_protection_client_template Settings	18-91
18-104	wss11_x509_token_with_message_protection_client_template Configuration Properties	18-92
18-105	wss11_x509_token_with_message_protection_service_template Configuration Properties	18-93
18-106	binding_oes_authorization_template Settings	18-94
18-107	binding_oes_authorization_template Configuration Properties	18-94
18-108	pii_security_template Settings	18-96
18-109	pii_security_template Configuration Properties	18-96
18-110	oracle/sts_trust_config_client_template Settings	18-96
18-111	oracle/sts_trust_config_client_template Properties	18-97
18-112	oracle/sts_trust_config_service_template Settings	18-98
18-113	oracle/sts_trust_config_service_template Properties	18-98
18-114	oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template Settings	18-99
18-115	oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template Properties	18-99
18-116	oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template Properties	18-100
18-117	oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template Settings	18-101
18-118	oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template Properties	18-102
18-119	oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template Properties	18-103
18-120	wss11_sts_issued_saml_with_message_protection_client_template Settings	18-104
18-121	oracle/wss11_sts_issued_saml_with_message_protection_client_template Properties	18-105
18-122	binding_authorization_template Settings	18-106
18-123	binding_authorization_template Properties	18-106
18-124	binding_permission_authorization_template Settings	18-107
18-125	binding_permission_authorization_template Properties	18-107
18-126	component_authorization_template Settings	18-107
18-127	component_authorization_template Properties	18-108
18-128	component_permission_authorization_template Settings	18-109
18-129	component_permission_authorization_template Properties	18-109
18-130	Supported Algorithm Suites	18-110
18-131	Request, Response, and Fault Message Signing and Encryption Settings	18-112
18-132	security_log_template Settings	18-113
18-133	security_log_template Properties	18-113

A-1	Security and Policy Annotations for Oracle Web Services	A-1
A-2	Attributes for javax.xml.ws.soap.Addressing Annotation	A-4
A-3	Attributes for com.oracle.webservices.api.tx.at.AtomicTransaction Annotation	A-5
A-4	Attributes for com.oracle.webservices.api.Buffering Annotation	A-6
A-5	Attributes for com.oracle.webservices.api.CacheBinaryContent Annotation	A-7
A-6	Attributes for oracle.webservices.annotations.async.CallbackManagementPolicy Annotation	A-8
A-7	Attributes for oracle.webservices.annotations.async.CallbackMtomPolicy Annotation	A-9
A-8	Attributes for oracle.wsm.metadata.annotation.CallbackPolicySet Annotation	A-10
A-9	Attributes for oracle.webservices.annotations.async.CallbackSecurityPolicy Annotation	A-11
A-10	Attributes for com.oracle.webservices.api.FastInfosetCallbackClient Annotation	A-12
A-11	Attribute for com.oracle.webservices.api.FastInfosetService Annotation	A-13
A-12	Attributes for oracle.webservices.annotations.ManagementPolicy Annotation	A-15
A-13	Attributes for com.oracle.webservices.api.MaxRequestSize Annotation	A-16
A-14	Attribute for com.oracle.webservices.api.MEXRequestProcessingService Annotation	A-17
A-15	Attribute for javax.xml.ws.soap.MTOM Annotation	A-18
A-16	Attribute for com.oracle.webservices.api.MTOMEncodeFaultService Annotation	A-18
A-17	Attributes for oracle.webservices.annotations.MtomPolicy Annotation	A-19
A-18	Attributes for oracle.webservices.annotations.Persistence Annotation	A-20
A-19	Attributes for oracle.wsm.metadata.annotation.PolicyReference Annotation	A-21
A-20	Attributes for oracle.wsm.metadata.annotation.PolicySet Annotation	A-22
A-21	Attribute for com.oracle.webservices.api.POXHttpBindingService Annotation	A-23
A-22	Attributes for oracle.wsm.metadata.annotation.Property Annotation	A-23
A-23	Attributes for oracle.webservices.annotations.ReliabilityPolicy Annotation	A-24
A-24	Attributes for oracle.webservices.annotations.ReliableMessaging Annotation	A-25
A-25	Attribute for oracle.webservices.annotations.RequestProcessingService Annotation	A-25
A-26	Attribute for oracle.webservices.annotations.SchemaValidation Annotation	A-26
A-27	Attributes for oracle.webservices.annotations.SecurityPolicy Annotation	A-27
A-28	Attributes for weblogic.wsee.jws.jaxws.owsm.SecurityPolicy Annotation	A-28
A-29	Attribute for oracle.webservices.annotations.SOAPRequestProcessingService Annotation	A-29
A-30	Attribute for oracle.webservices.annotations.TestPageProcessingService Annotation	A-29
A-31	Attribute for oracle.webservices.annotations.WSDLRequestProcessingService Annotation	A-30
A-32	Attribute for oracle.webservices.annotations.WSLoggingLevel Annotation	A-30
C-1	Oracle Extensions to WS-Policy Attributes	C-5
C-2	Attribute of <wsp:ExactlyOne> Element	C-6
C-3	Attributes of <orasp:Assertion> Element	C-8
C-4	Attributes of <orawsp:Config> Element	C-9
C-5	Attributes of <orawsp:PropertySet> Element	C-9

C-6	Attributes of <orawsp:Property> Element	C-10
C-7	Properties Used by the Predefined Assertions	C-10
C-8	Attributes of <orasp:sts-trust-config> Element	C-20
C-9	Attributes of <orasp:wss11-sts-issued-token-with-certificates> Element	C-30
C-10	Attributes of <orasp:wss-sts-issued-token-over-ssl> Element	C-34
C-11	Attributes of <orasp:attachment> Element	C-40
C-12	Attributes of <orasp:auth-header> Element	C-40
C-13	Attributes of <orasp:coreid-token> Element	C-41
C-14	Attributes of <orasp:element> Element	C-42
C-15	Attributes of <orasp:header> Element	C-43
C-16	Attributes of <orasp:issued-token> Element	C-44
C-17	Attributes of <orasp:kerberos-token> Element	C-44
C-18	Attributes of <orasp:msg-security> Element	C-45
C-19	Attributes of <orawsp:require-tls> Element	C-47
C-20	Attributes of <orasp:role> Element	C-48
C-21	Attributes of <orasp:saml-token> Element	C-48
C-22	Attributes of <orasp:username-token> Element	C-50
C-23	Attributes of <orasp:x509-token> Element	C-51
D-1	Attributes of Policy Set Element	D-1
D-2	Attributes of <wsp:policyReference> Element	D-2
E-1	OWSM Introspection Plug-in Parameter	E-2
E-2	OWSM Introspection Plug-in Supported Topologies	E-2
E-3	OWSM System Properties	E-4
E-4	OWSM Appliance User Properties - STS Trust	E-4
E-5	OWSM User Properties - Kerberos and SPNEGO	E-4
E-6	OWSM User Properties - wss11-sts-issued-token-with-certificates	E-5
E-7	OWSM Appliance User Properties - Configuration Bootstrapping	E-5

Preface

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

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 see [Understanding Web Services](#) to gain a better understanding of the two web service stacks supported in Oracle Fusion Middleware 12c.

The document is organized as follows:

- [Introduction to Oracle Web Services](#) introduces you to the concepts and tasks required to secure and administer web services. It provides a brief introduction to web service policies and helps you to decide which security policies are best for your web services. It also describes how to install and configure Oracle Web Services Manager on WebLogic Server.
- [Attaching and Managing Policies](#) describes how to work with policies and policy sets using Fusion Middleware Control and with WLST. Topics such as viewing, attaching, detaching, editing and overriding policies and policy sets are included.
- [Securing Web Services](#) describes how to configure your web service for message protection, SSL, authentication, and authorization. SAML and WS-Trust use cases are included as examples. You can also integrate hardware with Oracle Web Services Manager.
- [Managing and Troubleshooting Oracle Web Services Manager](#) describes how to diagnose problems and tune the performance of Oracle Web Services Manager. It also describes how to work with the Oracle Web Services Manager Repository.
- [Oracle Web Services Manager Predefined Policies and Assertions Templates](#) provides reference information describing the predefined policy and assertion templates provided with OWSM.
- [Security and Policy Reference for Oracle Web Services](#) provides reference information that describes web service security and policy annotations; assertion and policy set schemas; and the OWSM plug-in for Oracle Virtual Assembly Builder

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 Oracle Web Services Manager documents:

- *Administering Web Services*
- *Developing Extensible Applications for Oracle Web Services Manager*
- *Developing Fusion Web Applications with Oracle Application Development Framework*
- *Developing JAX-WS Web Services for Oracle WebLogic Server*
- *Developing Oracle Infrastructure Web Services*
- *Developing Applications with Oracle JDeveloper*
- *Developing SOA Applications with Oracle SOA Suite*
- *Interoperability Solutions Guide for Oracle Web Services Manager*
- *Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*
- *Securing WebLogic Web Services for Oracle WebLogic Server*
- *Understanding Oracle Web Services Manager*
- *Understanding WebLogic Web Services for Oracle WebLogic Server*
- *Understanding Web Services*
- *Use Cases for Securing Web Services Using Oracle Web Services Manager*
- *WebLogic Web Services Reference for Oracle WebLogic Server*

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.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in This Guide

Learn about the new and changed features of Oracle Web Services Manager (OWSM) that are described in this guide.

Topics:

- [New and Changed Features for 14c \(14.1.2.0.0\)](#)

New and Changed Features for 14c (14.1.2.0.0)

From an OWSM perspective, the new features include deployment of the OWSM Policy Manager (`wsm-pm`) across all managed servers and the Admin Server.

Part I

Introduction to Oracle Web Services

Learn about the concepts and tasks required to secure and administer web services.

Part I provides an introduction to web service policies and helps you to decide which security policies are best for your web services. It also describes how to install and configure Oracle Web Services Manager on WebLogic Server.

It includes the following chapters:

- [Overview of Oracle Web Services Security and Policy Management](#) , provides a brief overview of Oracle web services security and policy management using Oracle Web Services Manager.
- [Using Oracle Web Services Manager with Oracle WebLogic Server](#) , describes how to install and configure Oracle Web Services Manager on WebLogic Server.
- [Determining Which Predefined Policies to Use for a Web Service](#), provides a questionnaire to help you identify the security policies that best meet your requirements and a summary of predefined policies included in the current release.

1

Overview of Oracle Web Services Security and Policy Management

Learn about Oracle web services security and policy management using Oracle Web Services Manager (OWSM).

It includes the following topics:

- [Web Services Security and Policy Management](#)
- [Categories of Oracle Web Services Secured Using OWSM](#)

1.1 Web Services Security and Policy Management

Oracle Web Services Manager (OWSM) provides a policy framework to manage and secure web services consistently across your organization.

For details about OWSM, see *Understanding Oracle Web Services Manager*.

OWSM can be used by both developers, at design time, and system administrators in production environments:

- Application developers use Oracle JDeveloper to leverage the security and management features of the OWSM policy framework. For more information, see "Developing and Securing Web Services" in *Developing Applications with Oracle JDeveloper*.
- System administrators can leverage OWSM post-deployment using Oracle Enterprise Manager Fusion Middleware Control or the command line interface WebLogic Scripting Tool (WLST).

Details for using OWSM, including the predefined policies and assertions, to secure the web services in your environment are described throughout this document.

For definitions of unfamiliar terms found in this and other books, see the [Glossary](#).

1.2 Categories of Oracle Web Services Secured Using OWSM

Know more about the types and categories of Oracle web services you can secure using the OWSM framework.

The types and categories of Oracle web services are listed in [Table 1-1](#). For more information about the web service categories and the types of web services and clients, see "Overview of Web Services in Oracle Fusion Middleware 12c" in *Understanding Web Services*.

Table 1-1 Categories of Oracle Web Services Secured Using OWSM

Web Service Category	Web Service and Client Types
Oracle Infrastructure web services	Oracle ADF Services <ul style="list-style-type: none">• ADF Business Components services (SOAP and RESTful)• ADF web applications (SOAP and RESTful)• ADF data controls (SOAP only)

Table 1-1 (Cont.) Categories of Oracle Web Services Secured Using OWSM

Web Service Category	Web Service and Client Types
Oracle Infrastructure web services	Oracle Enterprise Scheduler <ul style="list-style-type: none"> • Web service jobs and callback services (SOAP only)
Oracle Infrastructure web services	Oracle Service Bus <ul style="list-style-type: none"> • Business and proxy services (SOAP, RESTful) • JCA adapters <p>Note: You can also attach a subset of OWSM policies to non-SOAP HTTP endpoints. For more information, see "Supported OWSM Seed Policies for WSDL (non-SOAP), XML, and Messaging Service Service Types with HTTP Transport" in <i>Developing Services with Oracle Service Bus</i>.</p>
Oracle Infrastructure web services	Oracle SOA web services <ul style="list-style-type: none"> • Service components (SOAP and RESTful) • Service and reference binding components (SOAP and RESTful) • JCA adapters
Java EE (WebLogic) web services	<ul style="list-style-type: none"> • JAX-WS (SOAP) web services • JAX-RS (RESTful) web services

2

Using Oracle Web Services Manager with Oracle WebLogic Server

You can install and configure Oracle Web Services Manager with Oracle WebLogic Server, to use a domain-wide administration port, modify the default user, and share OWSM Policy Managers between WebLogic Server domains.

It includes the following topics:

- [Installing Oracle Web Services Manager with WebLogic Server](#)
- [OWSM Configuration with a Domain-Wide Administration Port](#)
- [Cross-Component Wiring for Auto-Discovery of Policy Manager](#)
- [About Verifying Service Table Entries and Agent Bindings](#)
- [About Modifying the Default User](#)

2.1 Installing Oracle Web Services Manager with WebLogic Server

OWSM is installed by default when you install Oracle Fusion Middleware Infrastructure. However, if you have a standalone WebLogic Server environment with JAX-WS web services and clients deployed, you can install OWSM and use it to secure your web services and clients.

 **Note:**

OWSM is licensed only through SOA Suite; a standalone license is not available. For information about licensing, see [Licensing Considerations for Additional Features](#).

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

The Oracle Fusion Middleware Infrastructure standard installation topology includes OWSM, JRF and Enterprise Manager. Information about this topology and detailed installation procedures are provided in [Installing the Infrastructure Software](#). To use OWSM with WebLogic Server, follow the procedures in that document to install and configure the Oracle Fusion Middleware Infrastructure standard installation topology.

Once the installation is complete and the domain is configured, you can secure your Java EE (WebLogic) JAX-WS web services using OWSM, as provided in [Installing and Configuring the Oracle Fusion Middleware Infrastructure](#). However, OWSM policies cannot be applied to JAX-RPC web services.

After you complete the installation and configure the domain, you can secure your Java EE (WebLogic) JAX-WS web services using OWSM as described in this document. You cannot attach OWSM policies to JAX-RPC web services.

Procedures for administering Java EE (WebLogic) web services are provided in *Administering Web Services*.

2.2 Deploying Policy Manager

This topic lists the steps to deploy Policy Manager (PM) and Oracle Metadata Server (MDS) on all servers.



Note:

Policy manager can be targeted to any servers and clusters.

Deploying wsm-pm

Following are the steps to deploy wsm-pm on all the servers.

1. Connect WLS Remote Console to the WLS AdminServer.
2. Navigate to *wsm-pm* and open the **Targets** tab.
3. Select *wsm-pm* and click the **Change Targets** button.
4. Select all servers and clusters including AdminServer and click **Yes**.
5. Click **Activate Changes**.

Deploying mds-owsm

Following are the steps to deploy mds-owsm on all the servers.

1. Connect WLS Remote Console to the WLS AdminServer.
2. Navigate to *Services/DataSources/mds-owsm* and open the **Targets** tab.
3. Select all servers and clusters including AdminServer and click **Save**.
4. Click **Activate Changes**.

2.3 OWSM Configuration 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 OWSM 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 *Administering Server Environments for Oracle WebLogic Server*.
- "Configure the domain-wide administration port" in *Oracle WebLogic Server Administration Console Online Help*.

Configuring OWSM with a domain-wide administration port requires two main steps.

 **Note:**

Policy Manager can be targeted to any server or cluster within the environment. The following section describes the specific scenarios in which Policy Manager is targeted to the Admin Server.

1. [Targeting the Policy Manager to the Administration Server Using the WebLogic Remote Console](#)
2. [Specifying the policy accessor URL for the Policy Manager on the Administration Server Using Fusion Middleware Control](#)

2.3.1 Targeting the Policy Manager to the Administration Server Using the WebLogic Remote Console

The Policy Manager application is targeted to the Administration Server to configure OWSM. You can use WebLogic Remote Console to target the Policy Manager to the Administration Server.

Access the WebLogic Remote Console as described in [Accessing Oracle WebLogic Remote Console in Administering Web Services with Oracle Fusion Middleware](#).

1. You can also access the WebLogic Remote 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 in Administering Web Services with Oracle Fusion Middleware](#).
 - b. In the navigator pane, expand **WebLogic Domain** and select the domain for which you want to access the Remote Console.
The WebLogic Domain home page is displayed.
 - c. From the **WebLogic Domain** menu, select **WebLogic Server Remote Console**. Alternatively, you can click the **Oracle WebLogic Server Remote 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.

2.3.2 Specifying the policy accessor URL for the Policy Manager on the Administration Server Using Fusion Middleware Control

The policy accessor URL is specified for the Policy Manager on the Administration Server to configure OWSM. You can use Fusion Middleware Control to specify the policy accessor URL for the Policy Manager on the administration server.

To specify the policy accessor URL for the Policy Manager on the administration server:

1. Access the WSM Domain Configuration page, as described in "[Navigating to the WSM Domain Configuration Page](#)".
2. Select the **Policy Access** tab.
3. In the Policy Manager section of the page, clear the **Auto Discover** check box. The PM URL **Edit** button is enabled.
4. Click the PM URL **Edit** button.
5. In the Edit PM URL Values page, click the **+** sign and enter the 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 running 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 Policy Manager Problems Using the OWSM Policy Manager Page](#)".

6. Click **OK** to close the window.
7. Click **Apply** on the Policy Access page.

2.4 Cross-Component Wiring for Auto-Discovery of Policy Manager

Cross-component wiring provides a simplified method for wiring Fusion Middleware components. It automates the wiring process and provides the ability to diagnose wirings after they are established.

Wiring is simply a piece of configuration in one component that points to another component, such as a URL that points to the admin interface of a component. With cross-component wiring:

- Service providers publish their endpoints to a Service Table. These endpoints can be published automatically, such as when you create or extend a domain using the Configuration Wizard, or can be published manually by the administrator.
- Clients contain configuration that points to the service (for example, it has the service URL). The client is "bound" to the service by updating this configuration with the service information that was published to the Service Table. Binding is performed automatically

when creating or extending a domain using the Configuration Wizard, or can be done manually by the administrator.

When you install and configure Fusion Middleware on WebLogic Server, the local service table is created. It provides a means for service providers to publish endpoint information about their services, and for clients of these services to query and bind to these services. The Local Service table is scoped to a domain, and contains services that are offered by that domain. For detailed information about service tables and cross-component wiring, see "Cross-Component Wiring" in *Administering Oracle Fusion Middleware*.

OWSM uses cross-component wiring to auto-discover the Policy Manager in the domain. When you use the Configuration Wizard to create or update a domain that includes OWSM, Policy Manager URLs are published to the Local Service table. The OWSM Agent is automatically wired to the OWSM Policy Manager using the endpoint entries published to the Local Service table.

If, however, you change the domain using tools other than the Configuration Wizard (such as the WebLogic Remote Console, Fusion Middleware Control, or WLST), any changes to the Policy Manager URL are automatically published to the Local Service table but the OWSM Agent client is *not* automatically bound to the new URL. In this case, you need to manually bind the OWSM Agent to the Policy Manager URL. For more information, see "[Verifying Agent Bindings Using Fusion Middleware Control](#)".

You can change and disable the auto-discovery settings as described in the following sections:

- "[Understanding Configuring the Policy Manager Connection Using Fusion Middleware Control](#)"
- "[Configuring the Policy Manager Connection Using WLST](#)"

2.5 About Verifying Service Table Entries and Agent Bindings

Use Fusion Middleware Control to verify service table entries and agent bindings.

This topic describes how to verify service table entries and agent bindings using Fusion Middleware Control:

- [Verifying Service Table Entries and Components Using Fusion Middleware Control](#)
- [Verifying Agent Bindings Using Fusion Middleware Control](#)

2.5.1 Verifying Service Table Entries and Components Using Fusion Middleware Control

You can use the **Service Tables** page and the **Components** page in the Fusion Middleware Control to verify the service table entries and components.

To verify that the service table entry contains the correct URL for the Policy Manager:

- Using the **Service Tables** page
 1. From the **WebLogic Domain** menu, select **Cross Component Wiring**, then **Service Tables**.
 2. In the Service Table, verify that the URL in the Connection column and the Last Published date reflect the correct information for the OWSM Policy Manager with the Service ID `urn:oracle:fmw.owsm-pm:t3`.
 3. If you are using a static or dynamic cluster, then select the Service ID and click **Edit** and update the t3 and t3s values with the cluster name syntax.

The cluster name syntax is as follows:

```
cluster:t3://<cluster_name>
```

or

```
cluster:t3s://<cluster_name>
```

When you use `cluster:t3://<cluster_name>` or `cluster:t3s://<cluster_name>`, the CCW invocation fetches the complete list of members in the cluster at any given time, thus avoiding any dependencies on the initial servers and accounting for every member that is alive in the cluster then.

4. Click **OK**.
- Using the **Components** page
 1. From the **WebLogic Domain** menu, select **Cross Component Wiring**, then **Components**.
 2. In the Component Type table, select OWSM Policy Manager.
 3. In the Service End Points table, verify that the correct URL is shown in the Connection column and that the status is `Published` for the OWSM Policy Manager with Service ID `urn:oracle:fmw.owsm-pm:t3`. If the status is not `Published` (for example `Out of Sync`), you can also use this page to publish the Policy Manager connection. To do so, select the Policy Manager in the table and click **Publish**.

2.5.2 Verifying Agent Bindings Using Fusion Middleware Control

You can use the Fusion Middleware Control to verify that the OWSM Agent is wired correctly.

To verify that the OWSM Agent is wired correctly to the appropriate Policy Manager URL:

1. From the **WebLogic Domain** menu, select **Cross Component Wiring**, then **Components**.
2. In the Components Table, select **OWSM Agent**.
3. In the Client Configurations table, verify that the Client ID `owsm-pm-connection-t3` reflects the correct Policy Manager URL in the Connection column, and shows the status as `Wired` in the Status column.

If the Status column displays `Out of Sync`, you need to bind the Agent to the Policy Manager. To do so:

- a. Select `owsm-pm-connection-t3` in the Client Configurations table and click **Bind**.
- b. In the Bind Client Configuration page, verify that the Service End Point contains the correct Policy Manager URL and click **Yes**.

Confirmation is displayed on the Components page and the status of the Agent is changed to `Wired`.

2.6 About Modifying the Default User

The OWSM Agent run time uses the OracleSystemUser account and OracleSystemGroup, by default, to communicate to the server.

To modify the default user, perform the steps described in the following sections:

- [Configuring an Authentication Provider](#)

- [Configuring the Credential Store Provider](#)
- [Configuring the Policy Manager CSF Key for the Domain](#)
- [Modifying the User's Group or Role](#)
- [Determining that the User Has the Required Role](#)
- [Examples of Using OPSS Scripts to Manage Application Roles](#)

2.6.1 Configuring an Authentication Provider

This topic describes how to configure the authentication provider using Oracle WebLogic Server Remote Console.

To configure an authentication provider:

1. Configure an authentication provider, as described in "Configure Authentication and Identity Assertion providers" in *Oracle WebLogic Server Administration Console Online 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.

2.6.2 Configuring the Credential Store Provider

This topic describes how to configure the credential store provider.

Configure the credential store provider as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)" 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 (for example, `orcladmin` and `password`).

2.6.3 Configuring the Policy Manager CSF Key for the Domain

This topic describes how to configure the Policy Manager CSF key using Fusion Middleware Control.

To configure the Policy Manager CSF key for the domain:

1. Log into Fusion Middleware Control with the new default user account.
2. From the navigation pane, expand **WebLogic Domain** and select the domain to be configured.
3. From the **WebLogic Domain** menu, select **Web Services**, then **WSM Domain Configuration**.
4. Select the **Policy Accessor** tab.
5. Configure the Policy Manager CSF key as described in Step 3 in "[Understanding Configuring the Policy Manager Connection Using Fusion Middleware Control](#)".

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 "[Configuring the Credential Store Provider](#)".

Using the example provided in that section, select the OID key from the **PM Csf Key** drop-down menu, and enter `oracladmin` and `password` as the credential/password combination.

6. Click **Apply** and restart WebLogic Server.

2.6.4 Modifying the User's Group or Role

The OWSM Agent runtime uses the `OracleSystemUser` identity to access `wsm-pm`. If you define a new default user, it must be included in either the `Administrator` or `OracleSystemGroup` group (if the groups exist), or be mapped to the default OWSM logical roles

Map the new default user to the default OWSM logical roles defined in [Table 2-1](#) (if the groups do not exist).

Table 2-1 Default OWSM Logical Roles

Role	Default User	Permissions
<code>policy.Updater</code>	Administrators	Create, edit, delete, and update policies.
<code>policy.User</code>	All authenticated users	Read-only permission (for example, query/view document information).
<code>policy.Accessor</code>	<ul style="list-style-type: none"> Administrators OracleSystemGroup 	Used by the OWSM Policy Manager to secure EJBs that are accessed by the OWSM Agent runtime to attach policies.

2.6.5 Determining that the User Has the Required Role

The OWSM Agent run time uses the `OracleSystemUser` account and `OracleSystemGroup`, by default, to communicate to the server. After the default user is modified each User is assigned a required role.

To ensure that the user has the required role:

- If the Administrator or OracleSystemGroup groups exist in the LDAP or identity store, perform the following steps:
 1. In LDAP, add the user that you would like to use as a default administrative user.
 2. In WebLogic Server Remote Console, ensure that the user exists in the Administrator group. For more information, see "Manage Users and Groups" in *Oracle WebLogic Server Administration Console Online Help*.
- If the Administrator or OracleSystemGroup groups do not exist in the LDAP or identity store, you can manage application roles using one of the following OPSS scripts:
 - `grantAppRole`—Adds a principal (class and name) to a role with a given application stripe and name.
 - `revokeAppRole`—Removes a principal (class and name) to a role with a given application stripe and name.
 - `listAppRoleMembers`—Lists all members in a role with a given application stripe and role name.

For more information about these and other OPSS scripts, see "Managing the Policy Store" in *Securing Applications with Oracle Platform Security Services*.

2.6.6 Examples of Using OPSS Scripts to Manage Application Roles

The following examples illustrate how to use the OPSS scripts.

Before issuing the OPSS scripts, you must start WLST and connect to the running instance of WebLogic Server, as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.

The following command adds the `policy.Accessor` role to a principal named `PAPUser`:

```
grantAppRole(appStripe="wsm-pm",  
appRoleName="policy.Accessor",principalClass="weblogic.security.principal.WLSUserImpl",  
principalName="PAPUser")
```

The following command removes the `policy.Accessor` role from `OracleSystemGroup`:

```
revokeAppRole(appStripe="wsm-pm",  
appRoleName="policy.Accessor",principalClass="weblogic.security.principal.WLSGroupImpl",  
principalName="OracleSystemGroup")
```

The following command lists the members associated with the `policy.Accessor` role:

```
listAppRoleMembers(appStripe="wsm-pm", appRoleName="policy.Accessor")
```

3

Determining Which Predefined Policies to Use for a Web Service

Answer a questionnaire to help you determine which security policies are most appropriate for your web service. It also provides summaries of the predefined security policies included with the current release.

It includes the following topics:

- [Security Policy Questionnaire for a Web Service](#)
- [Summary of Predefined Security Policies for a Web Service](#)
- [OWSM Policies Supported for Java EE Web Services and Clients](#)
- [OWSM Policies Supported for RESTful Web Services and Clients](#)
- [OWSM Policies Supported for Web Services and Clients That Use SOAP Over JMS Transport](#)
- [OWSM Policies Supported for SOA Composite Services and Clients](#)
- [OWSM Policies that Require You to Configure SSL](#)
- [OWSM Policies Supported for Identity Context](#)
- [OWSM Policies Supported for WS-SecureConversation](#)
- [OWSM Policies Supported for JCA Adapters](#)
- [OWSM Policies Supported for OES Integration](#)
- [OWSM Policies Are Supported for PII](#)
- [OWSM Policies Supported for Oracle Service Bus](#)

3.1 Security Policy Questionnaire for a Web Service

The security policies that best meet your requirements is determined by the basic requirements, requirement for authentication, requirement for authentication and authorization, and requirement for authentication and message protection.

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 [Configuring Authorization Using Oracle Web Services Manager](#).
 - 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 "[Security Policies-Message Protection Only](#)".
 - 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, and Kerberos tokens. Authentication-only policies are described in [Authentication Only Policies](#).
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 [Configuring Authorization Using Oracle Web Services Manager](#) 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. Kerberos over SSL is also available via a custom policy.

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, X.509 certificates, and Kerberos.

For more information, see "[Security Policies-Messages Protection and Authentication](#)".

3.1.1 Choosing the Right Authentication Policy for a Web Service

OWSM includes many different authentication policies, and it might not be obvious which one best suites your needs. This topic describes selected authentication policies and when you might want to use them.

[Table 3-1](#) describes selected authentication policies and when you might want to use them. In [Table 3-1](#) the policy names are shown with wildcards (for example, `*username_token*`) to indicate all policies that have `username_token` in their name.)

Table 3-1 Choosing the Right Authentication Policy

Policy Type	Description
<code>*username_token*</code>	For these policies, the client needs to send the username and password to the web service. The password must be made available to the client in the credential store. This type of policy is useful for identity switching, in which a client needs to connect to a web service with an application identifier that is different from the actual end user name. It is the simplest of the authentication policies, and therefore compatible with the widest variety of third party clients.

Table 3-1 (Cont.) Choosing the Right Authentication Policy

Policy Type	Description
saml	<p>For these policies, the client needs to send a SAML assertion that contains the user name. There are variants of SAML, including the following:</p> <ul style="list-style-type: none"> • Sender Vouches. In this case the client constructs the SAML assertion. The server needs to be set up to trust the client. This policy is useful for identity propagation where a particular end user has already authenticated to the client, and the client needs to propagate this same user to the web service side, without having to know this user's password. Sender Vouches works best when communication between a middleware servers that are part of the same domain, or different domain that share the same credential store. Because they all share the same credential store and keys it is easier to make them all trust each other. Be cautious when using sender vouches for clients that are completely outside the domain. In sender vouches, the trust is based on the client's key, and with this key an attacker can impersonate any user. For example, do not use sender vouches from a client residing in an end user's desktop, because a malicious end user can easily get the client key, and with that be able to impersonate any other end user. • Holder Of Key from STS. The SAML holder of key is used in conjunction with a Secure Token Service (STS), which enables brokered trust. If there are many clients and many web services all in different unrelated security domains, it is difficult to make them all trust each other. Instead, they can trust a central entity, the STS. All the web services need to trust only the STS, and clients need to prove themselves to the STS by sending the credentials of the end users: user name, password, Kerberos tokens, and so forth.

Because SAML sender vouches and username token are among the most used policies, OWSM offers **OR group** policies combining these two, such as `oracle/wss_saml_or_username_token_service_policy`. In most situations web services should use this policy. This policy is also a perfect candidate for global policy attachment.

3.1.2 Choosing the Right Confidentiality and Integrity Policy for a Web Service

OWSM offers three levels of confidentiality and integrity.

The three levels of confidentiality and integrity are:

- **No confidentiality and integrity** — Confidentiality and integrity require cryptography, which consumes computing resources. In messages exchanged between middleware servers in a fire walled private network, there is no need to pay the price for confidentiality and integrity. The OWSM policies that do not have confidentiality and integrity do provide authentication through username token or SAML.
- **SSL based confidentiality and integrity** — SSL provides transport level confidentiality and integrity. With SSL you need to change your endpoints to use HTTPS, and make sure your clients talk to the HTTPS endpoints.

- Message Security based confidentiality and integrity — Message security offers much lower performance than SSL, but it has some advantages over SSL:
 - Unlike SSL, where the message stops being secure at the SSL termination point (which can be a load balancer, Oracle HTTP Server, or J2EE container) with message security the message remains secure all the way to the application.
 - With SSL, the security is at the container level. That is, all web services running on a container must share the same key. With message security, although the default is to share the same key throughout the domain, it is also possible to override the key on a per-Web-service basis.

There are two versions of message security offered in OWSM: wss10 and wss11. wss11 is an improvement over wss10 because every client does need to have its own client key, which is required for Wss10. (In certain policies such as SAML sender vouches, the client key is required in wss11 as well.)

wss11 is also faster because it requires fewer asymmetric key operations. However wss10 offers wider compatibility: some clients work with wss10 only.

Use wss11 policies unless you need to support a client that can use wss10 only.

3.2 Summary of Predefined Security Policies for a Web Service

Predefined security policies provide security for a Web Service. These policies are enforced either at the transport layer or SOAP header.

The following sections summarize the predefined security policies, based on the type of security they provide and whether the policy is enforced at the transport layer or SOAP header. For more information about the predefined policy categories, see "Policy Categories" in *Understanding Oracle Web Services Manager*. For full descriptions of the policies, see [Oracle Web Services Manager Predefined Policies](#).

- [Authentication Only Policies](#)
- [Message Protection Only Policies](#)
- [Message Protection and Authentication Policies](#)
- [Authorization Policies](#)
- [WS-Trust Policies](#)
- [MTOM Attachment Policies](#)
- [Reliable Messaging Policies](#)
- [No Behavior Policies](#)

3.2.1 Authentication Only Policies

This topic lists the authentication only policies provided for SOAP and RESTful web services.

[Table 3-3](#) summarizes the security policies that enforce authentication only for SOAP and RESTful web services.

Table 3-2 Authentication Only Policies—SOAP and RESTful Web Services

Client Policy	Service Policy	Authentication Transport
oracle/http_basic_auth_over_ssl_client_policy	oracle/http_basic_auth_over_ssl_service_policy	Yes

Table 3-2 (Cont.) Authentication Only Policies—SOAP and RESTful Web Services

Client Policy	Service Policy	Authentication Transport
N/A	oracle/http_oam_token_service_policy	Yes
oracle/http_saml20_token_bearer_client_policy	oracle/http_saml20_token_bearer_service_policy	Yes
oracle/http_saml20_token_bearer_over_ssl_client_policy	oracle/http_saml20_bearer_token_over_ssl_service_policy	Yes
Attach one of the following: <ul style="list-style-type: none"> oracle/wss_http_token_client_policy oracle/http_saml20_token_bearer_client_policy oracle/http_jwt_token_client_policy oracle/http_oauth2_token_client_policy To support HTTP OAM security, you must configure OAM Webgate to intercept the request. For more information, see " oracle/multi_token_rest_service_policy ".	oracle/multi_token_rest_service_policy	Yes
Attach one of the following: <ul style="list-style-type: none"> oracle/http_basic_auth_over_ssl_client_policy oracle/http_saml20_token_bearer_over_ssl_client_policy oracle/http_jwt_token_over_ssl_client_policy oracle/http_oauth2_token_identity_switch_over_ssl_client_policy oracle/http_oauth2_token_over_ssl_client_policy To support HTTP OAM security, you must configure OAM Webgate to intercept the request. For more information, see " oracle/multi_token_over_ssl_rest_service_policy ".	oracle/multi_token_over_ssl_rest_service_policy	Yes

[Table 3-3](#) 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 3-3 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_saml_token_client_policy	oracle/wss10_saml20_token_service_policy	No	Yes

Table 3-3 (Cont.) Authentication Only Policies—SOAP Web Services Only

Client Policy	Service Policy	Authentication Transport	Authentication SOAP
oracle/wss11_kerberos_token_client_policy	oracle/wss11_kerberos_token_service_policy	No	Yes

[Table 3-4](#) summarizes the security policies that enforce authentication only for AAuth2 and JWT web services.

Table 3-4 Authentication Only Policies—OAuth2 and JWT Web Services

Client Policy	Service Policy	Authentication Transport
oracle/http_oauth2_token_client_policy	oracle/http_jwt_token_service_policy	Yes
oracle/http_oauth2_token_identity_switch_opc_oauth2_over_ssl_client_policy		
oracle/http_oauth2_token_identity_switch_over_ssl_client_policy	oracle/http_jwt_token_over_ssl_service_policy	Yes
oracle/http_oauth2_token_opc_oauth2_client_policy	Reserved for use with Oracle Cloud.	
oracle/http_oauth2_token_opc_oauth2_over_ssl_client_policy		
oracle/http_oauth2_token_over_ssl_client_policy	oracle/http_jwt_token_over_ssl_service_policy	Yes
oracle/oauth2_config_client_policy	NA	
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_over_ssl_client_policy	oracle/http_jwt_token_over_ssl_service_policy	Yes

3.2.2 Message Protection Only Policies

This topic summarizes the policies that enforce message protection only, and indicates whether the policy is enforced at the transport layer or SOAP header.

Table 3-5 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

3.2.3 Message Protection and Authentication Policies

This topic 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 3-6 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: <ul style="list-style-type: none"> oracle/wss_saml_token_over_ssl_client_policy oracle/wss_username_token_over_ssl_client_policy 	oracle/wss_saml_or_username_token_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_token_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

Table 3-6 (Cont.) Message Protection and Authentication Policies

Client Policy	Service Policy	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss10_username_id_propagation_with_msg_protection_client_policy	oracle/wss10_username_id_propagation_with_msg_protection_service_policy	No	Yes	No	Yes
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: <ul style="list-style-type: none"> oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy 	oracle/wss11_saml_or_username_token_with_message_protection_service_policy	No	Yes	No	Yes
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_identity_switch_with_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

Table 3-6 (Cont.) Message Protection and Authentication Policies

Client Policy	Service Policy	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss11_x509_token_with_message_protection_client_policy	oracle/wss11_x509_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss_saml20_token_bearer_over_ssl_client_policy	oracle/wss_saml20_token_bearer_over_ssl_service_policy	No	Yes	No	Yes
oracle/wss_saml_token_bearer_client_policy	oracle/wss_saml_token_bearer_service_policy	No	Yes	No	Yes
oracle/wss_saml_token_bearer_identity_switch_client_policy	-	No	Yes	No	Yes
-	oracle/wss_saml_bearer_or_use_rname_token_service_policy	No	Yes	No	Yes

3.2.4 Authorization Policies

This topic summarizes the security policies that enforce authorization, and indicates whether the policy is enforced at the transport layer or SOAP header.

Table 3-7 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

3.2.5 WS-Trust Policies

This topic summarizes the WS-Trust policies.

Table 3-8 WS-Trust Policies

Client Policy	Service Policy	Authenticatio n Transport	Authenticatio n 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

3.2.6 MTOM Attachment Policies

This topic lists the MTOM Attachment policies supported in the current release.

- [oracle/wsmtom_policy](#).

Please note the following:

- If you configure MTOM from Fusion Middleware Control by attaching the [oracle/wsmtom_policy](#) policy (either via direct 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.

3.2.7 Reliable Messaging Policies

This topic lists the Reliable messaging policies supported in the current release.

- [oracle/no_reliable_messaging_policy](#)
- [oracle/wsrml0_policy](#)
- [oracle/wsrml1_policy](#)

3.2.8 No Behavior Policies

No behavior policies provide the ability to effectively disable a policy attached globally in a policy set. There are no configuration properties available for these policies. All of these policies use the same no behavior assertion.

Client Policies:

- [oracle/no_authentication_client_policy](#)
- [oracle/no_fast Infoset_client_policy](#)
- [oracle/no_jms_transport_client_policy](#)
- [oracle/no_messageprotection_client_policy](#)

Service Policies:

- [oracle/no_addressing_policy](#)
- [oracle/no_atomic_transaction_policy](#)
- [oracle/no_async_web_service_policy](#)
- [oracle/no_authentication_service_policy](#)
- [oracle/no_authorization_component_policy](#)
- [oracle/no_authorization_service_policy](#)
- [oracle/no_cache_binary_content_policy](#)
- [oracle/no_fast Infoset_service_policy](#)
- [oracle/no_jms_transport_service_policy](#)
- [oracle/no_max_request_size_policy](#)
- [oracle/no_messageprotection_service_policy](#)
- [oracle/no_mex_request_processing_service_policy](#)
- [oracle/no_mtom_policy](#)
- [oracle/no_mtom_encode_fault_service_policy](#)
- [oracle/no_persistence_policy](#)
- [oracle/no_pox_http_binding_service_policy](#)
- [oracle/no_reliable_messaging_policy](#)
- [oracle/no_request_processing_service_policy](#)
- [oracle/no_schema_validation_policy](#)
- [oracle/no_soap_request_processing_service_policy](#)
- [oracle/no_test_page_processing_service_policy](#)
- [oracle/no_ws_logging_level_policy](#)
- [oracle/no_wsdl_request_processing_service_policy](#)

3.3 OWSM Policies Supported for Java EE Web Services and Clients

All OWSM policies are not supported for Java EE web services and clients. Only a subset of OWSM policies are supported for Java EE web services and clients.

You can attach to WebLogic JAX-WS web services and clients the OWSM security policies in the following categories:

- Authentication only
- Message protection only
- Message protection and authentication
- Authorization
- WS-Trust
- WS-SecureConversation

OWSM policies in the following categories are not currently supported for WebLogic JAX-WS web services and clients:

- Atomic Transactions
- Configuration
- Management
- MTOM attachment
- No behavior
- Reliable messaging
- SOAP Over JMS Transport
- WS-Addressing policies

 **Note:**

- You can also secure Java EE (WebLogic) web services using WebLogic web service policies, which are provided by WebLogic Server. You manage WebLogic web service policies from the WebLogic Remote Console. For more information about the WebLogic web service policies, see *Using Oracle Web Services Manager Security Policies*
- A subset of WebLogic web service policies interoperate with OWSM policies. For more information, see "Interoperability with Oracle WebLogic Server 12c Web Service Security Environments" in *Interoperability Solutions Guide for Oracle Web Services Manager*.
- You cannot attach OWSM policies to JAX-RPC web services.

3.4 OWSM Policies Supported for RESTful Web Services and Clients

All OWSM policies are not supported for RESTful web services and clients. Only a subset of OWSM security policies are supported for RESTful web services and clients.

These policies are outlined in [Table 3-9](#).

 **Note:**

This section applies to Java EE, SOA, and Oracle Service Bus RESTful web services and clients.

You can attach OWSM policies to RESTful web services and clients that are built using Jersey 1.x JAX-RS RI only. RESTful web services and clients that are built using Jersey 2.5 JAX-RS RI cannot be secured using OWSM policies in this release. For more information about securing RESTful web services and clients built using Jersey 2.5 JAX-RS RI, see "Securing RESTful Web Services and Clients" in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*.

Table 3-9 OWSM Security Policies Supported for RESTful Web Services and Clients

Security	Supported Policies
Authentication Policies	Authentication policies defined in Table 3-2 .
Authorization	<ul style="list-style-type: none"> oracle/binding_authorization_denyall_policy oracle/binding_authorization_permitall_policy oracle/binding_permission_authorization_policy <p>Note: The oracle/binding_permission_authorization_policy permission-based policy is not supported for RESTful Oracle Service Bus web services and clients.</p>

 **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 "[Kerberos Configuration with SPNEGO Negotiation](#)".

3.5 OWSM Policies Supported for Web Services and Clients That Use SOAP Over JMS Transport

All OWSM policies are not supported for web services and clients that use SOAP over JMS transport. Only a subset of OWSM security policies are supported for web services and clients that use SOAP over JMS transport.

These supported policies include:

- wsmtom_policy
- wss_saml_token_bearer_client_policy
- wss_username_token_client_policy and wss_username_token_service_policy
- wss10_message_protection_client_policy and wss10_message_protection_service_policy
- wss10_saml_token_client_policy and wss10_saml_token_service_policy
- wss10_saml_hok_token_with_message_protection_client_policy and wss10_saml_hok_token_with_message_protection_service_policy
- wss10_saml_token_with_message_integrity_client_policy and wss10_saml_hok_token_with_message_integrity_service_policy
- wss10_saml_token_with_message_protection_client_policy and wss10_saml_token_with_message_protection_service_policy
- wss10_saml_token_with_message_protection_ski_basic256_client_policy and wss10_saml_token_with_message_protection_ski_basic256_service_policy
- wss10_username_token_with_message_protection_client_policy and wss10_username_token_with_message_protection_service_policy
- wss10_x509_token_with_message_protection_client_policy and wss10_x509_token_with_message_protection_service_policy
- wss11_kerberos_token_client_policy and wss11_kerberos_token_service_policy
- wss11_kerberos_token_with_message_protection_client_policy and wss11_kerberos_token_with_message_protection_service_policy
- wss11_kerberos_token_with_message_protection_basic128_client_policy and wss11_kerberos_token_with_message_protection_basic128_service_policy
- wss11_message_protection_client_policy and wss11_message_protection_service_policy
- wss11_saml_token_identity_switch_with_message_protection_client_policy
- wss11_saml_token_with_message_protection_client_policy and wss11_saml_token_with_message_protection_service_policy
- wss11_x509_token_with_message_protection_client_policy and wss11_x509_token_with_message_protection_service_policy
- wss11_x509_token_with_message_protection_wssc_client_policy and wss11_x509_token_with_message_protection_wssc_service_policy
- wss11_x509_token_with_message_protection_wssc_reauthn_client_policy and wss11_x509_token_with_message_protection_wssc_reauthn_service_policy
- wss11_sts_issued_saml_hok_with_message_protection_client_policy and wss11_sts_issued_saml_hok_with_message_protection_service_policy
- wss11_username_token_with_message_protection_client_policy and wss11_username_token_with_message_protection_service_policy
- wss11_username_token_with_message_protection_wssc_client_policy and wss11_username_token_with_message_protection_wssc_service_policy

3.6 OWSM Policies Supported for SOA Composite Services and Clients

You can attach various OWSM policies for SOAP SOA composite service and clients, but only a subset of OWSM security policies are supported for RESTful web services and clients.

For SOAP SOA composite service and clients, all policies described in "[Oracle Web Services Manager Predefined Policies](#)" apply except the configuration policies, described in "[Configuration Policies](#)".

For RESTful SOA composite services and clients, see "[OWSM Policies Supported for RESTful Web Services and Clients](#)".

3.7 OWSM Policies that Require You to Configure SSL

This topic lists the OWSM policies that require you to configure SSL and the templates that can be used to create these policies.

Refer to the following sections for more details:

- [List of Policies That Require You to Configure SSL](#)
- [List of Templates to Create Policies that Require SSL](#)
- [List of Policies That Require You to Configure Two-Way SSL](#)
- [List of Templates to Create Policies that Require Two-way SSL](#)

3.7.1 List of Policies That Require You to Configure SSL

The OWSM 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_username_token_over_ssl_service_policy`
- `oracle/wss_username_token_over_ssl_client_policy`
- `http_basic_auth_over_ssl_client_policy`
- `http_basic_auth_over_ssl_service_policy`
- `http_jwt_token_over_ssl_client_policy`
- `http_jwt_token_over_ssl_service_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_over_ssl_client_policy`

- `http_oauth2_token_over_ssl_client_policy`
- `http_saml20_token_bearer_over_ssl_client_policy`
- `http_saml20_token_bearer_over_ssl_service_policy`
- `multi_token_over_ssl_rest_service_policy`
- `wss_http_token_over_ssl_client_policy`
- `wss_http_token_over_ssl_service_policy`
- `wss_saml20_token_bearer_over_ssl_client_policy`
- `wss_saml20_token_bearer_over_ssl_service_policy`
- `wss_saml20_token_over_ssl_client_policy`
- `wss_saml20_token_over_ssl_service_policy`
- `wss_saml_or_username_token_over_ssl_service_policy`
- `wss_saml_token_bearer_over_ssl_client_policy`
- `wss_saml_token_bearer_over_ssl_service_policy`
- `wss_saml_token_over_ssl_client_policy`
- `wss_saml_token_over_ssl_service_policy`
- `wss_sts_issued_saml_bearer_token_over_ssl_client_policy`
- `wss_sts_issued_saml_bearer_token_over_ssl_service_policy`
- `wss_username_token_over_ssl_client_policy`
- `wss_username_token_over_ssl_service_policy`
- `wss_username_token_over_ssl_wssc_client_policy`
- `wss_username_token_over_ssl_wssc_service_policy`

3.7.2 List of Templates to Create Policies that Require SSL

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`
- `http_jwt_token_over_ssl_client_template`
- `http_jwt_token_over_ssl_service_template`

- `http_oauth2_token_over_ssl_client_template`
- `wss11_kerberos_token_over_ssl_client_template`
- `wss11_kerberos_token_over_ssl_service_template`
- `wss_http_token_over_ssl_client_template`
- `wss_http_token_over_ssl_service_template`
- `wss_saml20_token_bearer_over_ssl_client_template`
- `wss_saml20_token_bearer_over_ssl_service_template`
- `wss_saml20_token_over_ssl_client_template`
- `wss_saml20_token_over_ssl_service_template`
- `wss_saml_token_bearer_over_ssl_client_template`
- `wss_saml_token_bearer_over_ssl_service_template`
- `wss_saml_token_over_ssl_client_template`
- `wss_saml_token_over_ssl_service_template`
- `wss_sts_issued_saml_bearer_token_over_ssl_client_template`
- `wss_sts_issued_saml_bearer_token_over_ssl_service_template`
- `wss_username_token_over_ssl_client_template`
- `wss_username_token_over_ssl_service_template`

See [Oracle Web Services Manager Predefined Assertion Templates](#) and [Oracle Web Services Manager Predefined Policies](#) for more information on these assertions and policies.

3.7.3 List of Policies That Require You to Configure Two-Way SSL

This topic lists the OWSM policies that require you to configure two-way SSL.

- `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.

3.7.4 List of Templates to Create Policies that Require Two-way SSL

This topic lists the templates for creating a new OWSM policies that requires two-way SSL.

- `oracle/wss_saml_token_over_ssl_client_template`
- `oracle/wss_saml_token_over_ssl_service_template`

3.8 OWSM Policies Supported for Identity Context

All OWSM policies do not support the Identity Context feature. Only a subset of OWSM security policies are supported for the Identity Context feature.

Details about the Identity Context feature are described in "[About Propagating Identity Context with OWSM](#)".

The following SAML policies support the `propagate.identity.context` configuration property:

- 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

3.9 OWSM Policies Supported for WS-SecureConversation

Know more about the OWSM policies for which WS-SecureConversation is enabled by default.

The OWSM policies for which WS-SecureConversation is enabled by default are listed below:

- `oracle/wss11_saml_token_with_message_protection_wssc_client_policy`
- `oracle/wss11_saml_token_with_message_protection_wssc_service_policy`
- `oracle/wss11_saml_token_with_message_protection_wssc_reauthn_client_policy`
- `oracle/wss11_saml_token_with_message_protection_wssc_reauthn_service_policy`
- `oracle/wss11_username_token_with_message_protection_wssc_client_policy`
- `oracle/wss11_username_token_with_message_protection_wssc_service_policy`
- `oracle/wss11_x509_token_with_message_protection_wssc_client_policy`
- `oracle/wss11_x509_token_with_message_protection_wssc_service_policy`
- `oracle/wss_username_token_over_ssl_wssc_client_policy`
- `oracle/wss_username_token_over_ssl_wssc_service_policy`

In addition to these policies, policies based on many of the predefined assertion templates also support WS-SecureConversation. For more information, see [Oracle Web Services Manager Predefined Assertion Templates](#).

**Note:**

SOAP over JMS is not supported for WS-SecureConversation policies.

3.10 OWSM Policies Supported for JCA Adapters

All OWSM policies do not support JCA Adapters. Only a subset of OWSM security policy is supported for JCA adapters.

OWSM supports the following predefined policy for JCA adapters:

- `oracle/pii_security_policy`

In addition, custom policies that you create by cloning the `pii_security_policy` or that are based on the `oracle/pii_security_template` can also be used. For more information about using this policy, see [Protecting Personally Identifiable Information](#).

**Note:**

This policy is supported only for SOA and Oracle Service Bus environments.

3.11 OWSM Policies Supported for OES Integration

All OWSM policies do not support OES Integration. Only a subset of OWSM security policies are supported for OES Integration.

OWSM supports the following predefined policies for OES Integration:

- `oracle/binding_oes_authorization_policy`
- `oracle/binding_oes_masking_policy`
- `oracle/component_oes_authorization_policy`

In addition, custom policies that you create by cloning the OES policies or that are based on the OES templates can also be used. For more information about using these policies, see "[About Configuring Fine-Grained Authorization Using Oracle Entitlements Server](#)"

3.12 OWSM Policies Are Supported for PII

All OWSM policies do not support Personally Identifiable Information (PII). Only a subset of OWSM security policies are supported for PII.

OWSM supports the following predefined policy for protecting PII:

- `oracle/pii_security_policy`

In addition, custom policies that you create by cloning the `pii_security_policy` or that are based on the `oracle/pii_security_template` can also be used. For more information about using this policy, see [Protecting Personally Identifiable Information](#).



Note:

This policy is supported only for SOA and Oracle Service Bus environments.

3.13 OWSM Policies Supported for Oracle Service Bus

Know more about the supported OWSM policies for Oracle Service Bus.

For Oracle Service Bus, all policies described in "[Oracle Web Services Manager Predefined Policies](#)" apply except those specified in [Table 3-10](#). The table lists unsupported OWSM assertions for both SOAP and non-SOAP services, shows which policies contain the assertions, and describes the affected capabilities and alternatives to achieve the capabilities. Any assertions not listed are supported, including user-defined assertions.

Table 3-10 Unsupported Policies and Assertions for Oracle Service Bus

Unsupported Assertion	OWSM Policies Containing the Assertion	Capability Affected and Alternative
binding-permission-authorization	<code>oracle/binding_permission_authorization_policy</code>	Permission-based access control to service. Alternative: Use XACML authorization policies.

Table 3-10 (Cont.) Unsupported Policies and Assertions for Oracle Service Bus

Unsupported Assertion	OWSM Policies Containing the Assertion	Capability Affected and Alternative
sca-component-authorization	oracle/ component_authorization_denyall_ policy oracle/ component_authorization_permitall_ policy	Role-based access control to deny/permit all to access the component. Alternative: Not applicable
sca-component-permission-authorization	oracle/ component_permission_authorization_ policy	Permission based Access Control to component Alternative: Not applicable
OptimizedMimeSerialization	oracle/wsmtom_policy	Message Transmission Optimization Mechanism (MTOM) Alternative: Use MTOM configuration directly on proxy/business service.
RM Assertion	oracle/reliable_messaging_policy oracle/wsrml0_policy oracle/wsrml1_policy	WS-RM 1.0/1.1 Alternative: Use the WS transport directly in Service Bus for WS-RM 1.0.
UsingAddressing	oracle/wsaddr_policy	To require WS-Addressing Alternative: Configure WS-Addressing on business services that use the SOA-DIRECT transport; or add WS-Addressing to messages in a Service Bus pipeline.

Part II

Attaching and Managing Policies

Learn how to work with policies and policy sets using Fusion Middleware Control and WLST. Topics such as viewing, attaching, detaching, editing and overriding policies and policy sets are included.

Part II contains the following chapters:

- [Attaching Policies to Manage and Secure Web Services](#), describes how to use Fusion Middleware Control to attach security policies to web services.
- [Overriding Policy Configuration Properties](#), describes how to override security policy configuration properties by using Fusion Middleware Control and WLST.
- [Managing Web Service Policies with Fusion Middleware Control](#), describes how to work with web service security policies. Topics include creating, generating, editing, importing, and exporting policies. It also describes how to work with assertion templates.

4

Attaching Policies to Manage and Secure Web Services

Security policies provide a framework to manage and secure web services consistently across your organization.

Security policies can be attached directly to policy subjects, such as web services or clients. Policy sets provide a means to attach policies globally to a range of endpoints of the same type. This topic also describes how to attach policies at design time, and how to attach and manage policies and policy sets using Oracle Enterprise Manager Fusion Middleware Control and the command-line interface WebLogic Scripting Tool (WLST) post-deployment.

It includes the following topics:

- [Overview of Policy Attachment](#)
- [Understanding Attaching Policies to Web Services and Clients at Design Time](#)
- [About Attaching Policies to Web Services and Clients Using Fusion Middleware Control](#)
- [About Attaching Policies to Web Services and Clients Using WLST](#)
- [About Attaching Policies to Servlet Applications](#)
- [About Securing the URI patterns for Resources in RESTful Web Services](#)
- [Run-time Constraints in Policy Sets](#)
- [About Defining the Type and Scope of Resources for Globally Attached Policies](#)
- [Migrating Direct Policy Attachments to Global Policy Attachments](#)
- [Disabling a Globally Attached Policy](#)
- [Specifying the Priority of a Policy Attachment](#)
- [Managing Endpoint Configuration Properties Using Fusion Middleware Control](#)
- [Determining the Secure Status of an Endpoint](#)
- [How the Effective Set of Policies is Calculated](#)
- [Determining the Source of Policy Attachments](#)

4.1 Overview of Policy Attachment

A policy subject is the target resource to which OWSM policies are attached. There are different policies for different types of resources (for example, a web service or client or a SOA component).

For more information about the policy subjects with which policies can be associated, see "Attaching Policies to Policy Subjects" in *Understanding Oracle Web Services Manager*.

There are two ways to attach policies to web service clients and web services: at the client and service design time, and post deployment.

- At design time, you can attach OWSM security and management policies to applications programmatically. You typically do this using your favorite IDE, such as Oracle JDeveloper. Oracle JDeveloper automates ADF and SOA client policy attachment.
- Post-deployment, you attach security and management policies to Oracle Infrastructure web services and WebLogic web services using Oracle Enterprise Manager Fusion Middleware Control or WLST. This provides the most power and flexibility because it moves web service security to the control of the security administrator. Policies can be attached directly to an endpoint, or globally to a range of endpoints using policy sets.

 **Note:**

Globally attached policies (using policy sets) are supported for RESTful web services and clients, Oracle Infrastructure web services and clients, and Java EE web services and clients. However, if a policy set includes non-security policies, those non-security policies are ignored and therefore not included in the effective policy sets calculated for Java EE web services and clients.

Globally attached policies are not supported for standalone Java EE clients.

You can attach OWSM policies to RESTful web services and clients that are built using Jersey 2.x JAX-RS RI only. For more information about securing RESTful web services and clients built using Jersey 2.x JAX-RS RI, see "Securing RESTful Web Services and Clients" in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*.

Regardless of whether you attach a policy at design time or post-deployment, 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 invocation of the web service operation returns an error.

For example, since the `oracle/wss_http_token_over_ssl_service_policy` policy requires one-way authentication, the client policy must also be set for one-way 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 and Editing Web Service Policies](#)" increases the likelihood that the client policy will work with the service policy.

 **Note:**

If the OWSM Policy Manager is not available, any modifications that you make to policy attachments for ADF and WebCenter services and clients will not be saved to the OWSM repository. For information about troubleshooting the connection to the OWSM Policy Manager, see "[Diagnosing Policy Manager Problems Using the OWSM Policy Manager Page](#)".

4.2 Understanding Attaching Policies to Web Services and Clients at Design Time

OWSM security and management policies are attached to web services and clients using annotations based on whether you are creating an Oracle Infrastructure web service or Java EE (WebLogic) web service or client.

You can find details about policy attachment in the following sections:

- [About Attaching Policies to Java EE Web Services and Clients at Design Time](#)
- [About Attaching Policies to RESTful Web Services and Clients at Design Time](#)
- [About Attaching Policies to Oracle Infrastructure Web Services and Clients at Design Time](#)

4.2.1 About Attaching Policies to Java EE Web Services and Clients at Design Time

You can attach policies to Java EE web services and clients at design time, as described in the following topics:

- [Attaching Policies to Java EE Web Services and Clients Using Annotations](#)
- [Attaching Policies to Java EE Web Service Clients Using Feature Classes](#)
- [Attaching Policies to Java EE Web Services and Clients Using JDeveloper](#)

4.2.1.1 Attaching Policies to Java EE Web Services and Clients Using Annotations

To attach security policies to Java EE web services and clients, you can use one of the following annotations:

- `weblogic.wsee.jws.jaxws.owsm.SecurityPolicy` (single policy)
- `weblogic.wsee.jws.jaxws.owsm.SecurityPolicies` (multiple policies)

Only a subset of OWSM policies are supported for Java EE web services. For more information, see [OWSM Policies Supported for Java EE Web Services and Clients](#)

For Java EE web services, you can attach OWSM security policies at the class level only. For Java EE web service clients, you can attach OWSM policies to web service injection targets, defined using the `java.xml.ws.WebServiceRef` annotation.

When attaching security policies to web service clients, you can override a policy configuration property using the `weblogic.wsee.jws.jaxws.owsm.Property` annotation with the `@SecurityPolicy` annotation, as shown in [#unique_239/unique_239_Connect_42_CIHDFCDA](#) below. In addition, you can override a policy configuration property using the JAX-WS `RequestContext`, as described in [About Overriding Client Policy Configuration Properties at Design Time](#).

 **Note:**

Attaching OWSM security policies to Java EE web service clients using Feature classes takes precedence over annotations. For more information, see [Attaching Policies to Java EE Web Service Clients Using Feature Classes](#).

For more information about:

- The `@SecurityPolicy`, `@SecurityPolicies`, and `@Property` annotations, see "WebLogic-specific Annotations" in *WebLogic Web Services Reference for Oracle WebLogic Server*.
- Developing Java EE web services, see "Developing JAX-WS Web Services" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.
- Developing Java EE web service clients and defining web service injection targets using `@WebServiceRef`, see "Developing Java EE Web Service Clients" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.

The following example shows how to attach multiple OWSM security policies to a Java EE web service using annotations.

```
import javax.ws.WebService;
import weblogic.wsee.jws.jaxws.owsm.SecurityPolicy;
import weblogic.wsee.jws.jaxws.owsm.SecurityPolicies;
...
@SecurityPolicies({
    @SecurityPolicy(uri=
        "policy:oracle/wss10_username_token_with_message_protection_server_policy"),
    @SecurityPolicy(uri="policy:oracle/authorization_policy")
})
@WebService
public class HelloWorldImpl { ... }
```

The following example shows how to attach a policy to a Java EE web service client using annotations. In this example, the `@Property` annotation is used to override the keystore recipient alias configuration property when attaching the client policy.

??

```
package wsrn_jaxws.example;
import java.xml.ws.WebService;
import java.xml.ws.WebServiceRef;
import weblogic.wsee.jws.jaxws.owsm.SecurityPolicy;
import weblogic.wsee.jws.jaxws.owsm.SecurityPolicies;
import oracle.wsm.security.util.SecurityConstants.ClientConstants;
...
@WebServiceRef(name="MyServiceRef")
@SecurityPolicies({
    @SecurityPolicy(uri="policy:oracle/wss10_message_protection_client_policy",
        properties = {
            @Property(name="ClientConstants.WSS_KEYSTORE_LOCATION",
                value="c:/mykeystore.jks")
        }
    ),
    @SecurityPolicy(uri="policy:oracle/authorization_policy")
})
Service service;
...

```


4.2.1.2 Attaching Policies to Java EE Web Service Clients Using Feature Classes

You can use one of the following Feature classes to attach OWSM security policies to Java EE web service clients:

Note:

If you attach OWSM policies using Feature classes at design time, you will not be able to add or modify the policies using Fusion Middleware Control after the client application is deployed. If you wish to be able to add or modify policies post-deployment, it is recommended that you use one of the following policy attachment methods:

- Use annotations to attach OWSM policies to a web service client at design time, as described in [Attaching Policies to Java EE Web Services and Clients Using Annotations](#).
- Use Fusion Middleware Control to attach OWSM policies to a web service client post-deployment, as described in [Attaching Policies Directly to Web Service Clients Using Fusion Middleware Control](#).

- `weblogic.wsee.jws.jaxws.owsm.SecurityPolicyFeature` class (single policy)
- `weblogic.wsee.jws.jaxws.owsm.SecurityPoliciesFeature` class (multiple policies)

Only a subset of OWSM policies are supported for Java EE web services. For more information, see [OWSM Policies Supported for Java EE Web Services and Clients](#)

Note:

Attaching OWSM policies using Feature classes takes precedence over annotations (described in [Attaching Policies to Java EE Web Services and Clients Using Annotations](#)).

The following example shows how to use the `SecurityPolicyFeature` class to attach an OWSM policy to a web service client.

Note:

The `oracle/wss_username_token_client` policy shown in this example 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`.

```
...
JAXWSService jaxWsService = new JAXWSService ();
weblogic.wsee.jws.jaxws.owsm.SecurityPolicyFeature securityFeature = new
weblogic.wsee.jws.jaxws.owsm.SecurityPolicyFeature {
```

```

new weblogic.wsee.jws.jaxws.owsm.SecurityPolicyFeature("policy:oracle/
wss_username_token_client_policy" );

JAXWSServicePort port = jaxWsService.getJaxWsServicePort(securityFeature);
...

```

The following example shows how to use the `SecurityPoliciesFeature` class to attach multiple OWSM policy to a web service client.

```

...
weblogic.wsee.jws.jaxws.owsm.SecurityPoliciesFeature
securityFeature = new weblogic.wsee.jws.jaxws.owsm.SecurityPoliciesFeature
{
new weblogic.wsee.jws.jaxws.owsm.SecurityPolicyFeature(
    new String[] {"policy:oracle/wss_username_token_client_policy",
        "policy:oracle/authorization_policy"});
...

```

4.2.1.3 Attaching Policies to Java EE Web Services and Clients Using JDeveloper

For information about attaching policies to Java EE web services and client using JDeveloper, see "How to Attach Policies to JAX-WS Web Services and Clients" in *Developing Applications with Oracle JDeveloper*.

4.2.2 About Attaching Policies to RESTful Web Services and Clients at Design Time

You can attach policies to RESTful web services and clients at design time, as described in the following topics:

- [Attaching Policies to RESTful Web Services Using Annotations](#)
- [Attaching Policies to RESTful Web Service Clients Using Feature Classes](#)
- [Attaching Policies to RESTful Web Services and Clients Using JDeveloper](#)

Note:

You can attach OWSM policies to RESTful web services and clients that are built using Jersey 2.x JAX-RS RI only. For more information about securing RESTful web services and clients built using Jersey 2.x JAX-RS RI, see "Securing RESTful Web Services and Clients" in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*.

If you have modified the `web.xml` deployment descriptor file to define the OWSM servlet filter in order to attach policies to a servlet application, as described in "[About Attaching Policies to Servlet Applications](#)", the deployment descriptor definition takes precedence over the policy attachment procedures described in this section.

4.2.2.1 Attaching Policies to RESTful Web Services Using Annotations

To attach security policies to RESTful web services, you can use one of the following annotations, included in the `oracle.wsm.metadata.annotation` package:

 **Note:**

You can attach OWSM policies to RESTful web services and clients that are built using Jersey 2.x JAX-RS RI. Jersey1.x client form 12.1.3 is also supported.

For more information about securing RESTful web services and clients built using Jersey 2.x JAX-RS RI, see "Securing RESTful Web Services and Clients" in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*.

- @PolicySet
- @PolicyReference
- @Property

Only a subset of OWSM policies are supported for RESTful web services. For more information, see [OWSM Policies Supported for RESTful Web Services and Clients](#)

You can attach policies programmatically to the JAX-RS Application class only.

When attaching security policies to RESTful web services, you can override a policy configuration property using the @Property annotation with the @PolicyReference annotation, as shown in the example below.

For more information about the annotations, see [Security and Policy Annotations for Oracle Web Services](#). For more information about the predefined policies, see [Oracle Web Services Manager Predefined Policies](#).

The following example shows to secure a JAX-RS Application class using the @PolicySet, @PolicyReference, and @Property annotations.

```
...
import javax.ws.rs.core.Application;
import javax.ws.rs.core.ApplicationPath;
import oracle.wsm.metadata.annotation.PolicySet;
import oracle.wsm.metadata.annotation.PolicyReference;
import oracle.wsm.metadata.annotation.Property;
...
@PolicySet(references = {
    @PolicyReference("oracle/wss_http_token_service_policy"),
    @PolicyReference(value = "oracle/binding_permission_authorization_policy",
        properties = {

@Property(name="resource",value="com.sun.jersey.samples.helloworld.resources.MyApplicatio
n"),
    @Property(name="action",value="")})
})
@ApplicationPath("resources")
public class MyApplication extends Application {
    public Set<Class<?>> getClasses() {
        Set<Class<?>> s = new HashSet<Class<?>>();
        s.add(HelloWorldResource.class);
        return s;
    }
}
```

4.2.2.2 Attaching Policies to RESTful Web Service Clients Using Feature Classes

 **Note:**

You can attach OWSM policies to RESTful web services and clients that are built using Jersey 2.x JAX-RS RI. For more information about securing RESTful web services and clients built using Jersey 2.x JAX-RS RI, see "Securing RESTful Web Services and Clients" in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*.

It is recommended that you use jersey2.x for building client but jersey 1.x client available in 12.1.3 is also supported. The plain JEE clients created using Jersey1.x/2.x are not managed clients, and therefore, Oracle Enterprise Manager cannot be used to attach policies to these. These have to be created programmatically by using the feature classes.

If you attach OWSM policies using Feature classes at design time, you will not be able to add or modify the policies using Fusion Middleware Control after the client application is deployed. If you wish to be able to add or modify policies post-deployment, it is recommended that you use Fusion Middleware Control to attach OWSM policies to a web service client post-deployment, as described in [Attaching Policies Directly to Web Service Clients Using Fusion Middleware Control](#).

You can programmatically attach OWSM security policies to RESTful web service clients using the Feature classes defined in [Table 4-1](#). The classes are provided in the `oracle.wsm.metadata.feature` package. For more information about the Feature classes, see [Table A-1](#).

Table 4-1 Feature Classes Used for Attaching Policies to RESTful Clients

Feature Class	Description
<code>AbstractPolicyFeature</code>	Base abstract class for policy subject feature classes.
<code>PolicySetFeature</code>	Set of policy references and configuration override properties to attach to the policy subject.
<code>PolicyReferenceFeature</code>	Single policy reference to attach to the policy subject.
<code>PropertyFeature</code>	Optional property that can be used to override the configuration of one or more policies.

When you create a RESTful client instance, optionally you can pass client configuration properties by defining a `org.glassfish.jersey.client.ClientConfig` and passing the information to the `create` method of the `javax.ws.rs.client.Client` class. Using the `ClientConfig`, you can attach an OWSM policy such as `oracle/http_jwt_token_client_policy` and override configuration properties.

The following is a sample client with jersey 2.x API:

```
package samples.helloworld.client;

import org.glassfish.jersey.client.ClientConfig;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation.Builder;
```

```

import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

import oracle.wsm.metadata.feature.PolicyReferenceFeature;
import oracle.wsm.metadata.feature.AbstractPolicyFeature;
import oracle.wsm.metadata.feature.PolicySetFeature;
import oracle.wsm.metadata.feature.PropertyFeature;
...
    public static void main(String[] args) {
        ClientConfig cc = new ClientConfig();
        cc.property(AbstractPolicyFeature.ABSTRACT_POLICY_FEATURE,
            new PolicySetFeature(
                new PolicyReferenceFeature(
                    "oracle/wss_http_token_client_policy", new
                    PropertyFeature(SecurityConstants.ConfigOverride.CO_CSF_KEY,
                        "weblogic-csf-key"))));
        Client client = ClientBuilder.newClient(cc);

        WebTarget webTarget = client.target("http://<host>:<port>/<context>/
<resource>")
        Builder request = webTarget.request("text/plain");

        String response = request.get(String.class);
        ...
        client.close();
    }

```

4.2.2.3 Attaching Policies to RESTful Web Services and Clients Using JDeveloper

For information about attaching policies to Java EE web services and client using JDeveloper, see "How to Attach Policies to RESTful Web Services and Clients" in *Developing Applications with Oracle JDeveloper*.

4.2.3 About Attaching Policies to Oracle Infrastructure Web Services and Clients at Design Time

You can attach policies to Oracle Infrastructure web services and clients at design time, as described in the following sections:

- [Attaching Policies to Oracle Infrastructure Web Services Using Annotations](#)
- [Attaching Policies to Oracle Infrastructure Web Service Clients Using Feature Classes](#)
- [Attaching Policies to Oracle Infrastructure Web Services Using Oracle JDeveloper](#)

4.2.3.1 Attaching Policies to Oracle Infrastructure Web Services Using Annotations

To attach policies to Oracle Infrastructure web services, you can use any of the annotations defined in [Security and Policy Annotations for Oracle Web Services](#).

For more information about the predefined policies that can be attached using the security and policy annotations, see [Oracle Web Services Manager Predefined Policies](#).

The following example shows how to attach policies to an asynchronous web service and to its callback client that will connect to the callback service.

```

...
import oracle.wsm.metadata.annotation.CallbackPolicySet;
import oracle.wsm.metadata.annotation.PolicySet;
import oracle.wsm.metadata.annotation.PolicyReference;

```

```

import oracle.wsm.metadata.annotation.Property;
...
@PortableWebService(serviceName = "EchoService", portName = "EchoPort")
@PolicySet(references = @PolicyReference(
    "oracle/wss_username_token_service_policy"))
@CallbackPolicySet(properties = @Property("reference.priority", "1"),
    references = {
        @PolicyReference("oracle/wss10_saml_token_client_policy"),
        @PolicyReference("oracle/log_policy") })
public class EchoService {
    public EchoService() {
        super();
    }

    public String echo(String message) {
        return message + " echoed";
    }
}

```

4.2.3.2 Attaching Policies to Oracle Infrastructure Web Service Clients Using Feature Classes

You can programmatically attach OWSM policies to Oracle Infrastructure web service clients using the Feature classes defined in [Table A-1](#).

Using the `RequestContext`, you can attach OWSM policies and override configuration properties. For example, the following code shows how to use the Feature classes in the `oracle.wsm.metadata.feature` package to attach the `oracle/wss_http_token_client_policy` policy to the client, and overrides the `CO_CSF_KEY` configuration property with the value `weblogic-csf-key`.

```

...
import oracle.wsm.metadata.feature.PolicyReferenceFeature;
import oracle.wsm.metadata.feature.AbstractPolicyFeature;
import oracle.wsm.metadata.feature.PolicySetFeature;
import oracle.wsm.metadata.feature.PropertyFeature;
...
void configureBasicAuth(Dispatch<SOAPMessage> dispatch) {
    Map<String, Object> reqContext = dispatch.getRequestContext();
    reqContext.put(AbstractPolicyFeature.ABSTRACT_POLICY_FEATURE,
        new PolicySetFeature(
            new PolicyReferenceFeature(
                "oracle/wss_http_token_client_policy"), new
                PropertyFeature(SecurityConstants.ConfigOverride.CO_CSF_KEY,
                    "weblogic-csf-key")));
}
...

```

4.2.3.3 Attaching Policies to Oracle Infrastructure Web Services Using Oracle JDeveloper

When developing an application using JDeveloper, you can take advantage of the wizards available to attach policies to Oracle Infrastructure web services and clients. For more information, see:

- "Managing Policies" and "Attaching Policies to Binding Components and Service Components" in *Developing SOA Applications with Oracle SOA Suite*.

- "Securing Web Service Data Controls" in *Application Development Framework Developer's Guide*
- "Developing and Securing Web Services" in the *Developing Applications with Oracle JDeveloper*.

4.3 About Attaching Policies to Web Services and Clients Using Fusion Middleware Control

OWSM policies can either be attached to a single policy subject or to multiple subjects (global attachment) using Fusion Middleware Control.

You can find in these sections, distribution of how to attach policies to a single policy subject and to multiple subjects (global attachment) and how to validate the subject once policies are attached:

- [Attaching Policies Directly Using Fusion Middleware Control](#)
- [About Attaching Policies Globally Using Fusion Middleware Control](#)
- [Viewing Policies Attached to a Web Service Using Fusion Middleware Control](#)
- [Validating Policy Attachments](#)
- [About Validating a Policy Set](#)

4.3.1 Attaching Policies Directly Using Fusion Middleware Control

Learn more about how to attach and detach policies to and from a single subject and to validate the subject once policies are attached:

- ["Attaching Policies Directly to a Single Subject Using Fusion Middleware Control"](#)
- ["Attaching Policies Directly to Web Service Clients Using Fusion Middleware Control"](#)
- ["Enabling or Disabling Directly Attached Policies Using Fusion Middleware Control"](#)
- ["Detaching Directly Attached Policies Using Fusion Middleware Control"](#)

Please note:

- For WebLogic Java EE web services:
 - Only a subset of OWSM security policies can be attached. For more information, see [OWSM Policies Supported for Java EE Web Services and Clients](#)
 - OWSM policies and WebLogic web service policies cannot be attached to the same endpoint. If a Java EE endpoint has any WebLogic policies attached, the WSM Policies tab is not displayed and you cannot attach OWSM security policies. Instead, the WebLogic Policy Violations tab is displayed showing violation details about the WebLogic web service policies attached to the endpoint.

Note that WebLogic policies can be attached using the WebLogic Server Remote Console. You cannot attach WebLogic policies using Fusion Middleware Control.
- For RESTful web services:
 - You can attach OWSM policies to RESTful web services and clients that are built using Jersey 2.x JAX-RS RI only. For more information about securing RESTful web services and clients built using Jersey 2.x JAX-RS RI, see "Securing RESTful Web Services and Clients" in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*.

- Only a subset of OWSM security policies can be attached. For more information, see [OWSM Policies Supported for RESTful Web Services and Clients](#)
- If you have modified the `web.xml` deployment descriptor file to define the OWSM servlet filter in order to attach policies to a servlet application, as described in "[About Attaching Policies to Servlet Applications](#)", the deployment descriptor definition takes precedence over the policy attachment procedures described in this section.
- For SOA composite services, only a subset of OWSM policies apply. For more information, see "[OWSM Policies Supported for SOA Composite Services and Clients](#)".

4.3.1.1 Attaching Policies Directly to a Single Subject Using Fusion Middleware Control

A **subject** is an entity to which a policy can be associated. You can attach one or more policies to a subject.

Note:

When attaching policies directly to subjects within a clustered server environment, the policy attachment details will be propagated to other servers in the cluster following a brief delay. To expedite the propagation of the information, perform one of the following steps:

- Restart the other servers in the cluster.
- Configure the cache refresh properties to minimize the delay. For more information, see "[High Availability Configuration and Cache Management Using Fusion Middleware Control](#)".

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.

For more information, see "How Policies are Executed" in *Understanding Oracle Web Services Manager*.

To attach a policy directly to a single subject:

1. View the details for a web service endpoint, as described in "Viewing the Details for a Web Service Endpoint Using Fusion Middleware Control" in *Administering Web Services*.

Note:

When attaching policies to RESTful web services, you can attach policies to RESTful applications only, as described in "Viewing the Details for a RESTful Service Application" in *Administering Web Services*. You cannot attach policies to RESTful resources.

2. Click the **Attach/Detach Policies** link at the top of the page.

The WSM Policy Subject Configuration page is displayed and any policies that are already globally and directly attached to the endpoint are displayed. For example, see [Figure 4-1](#).

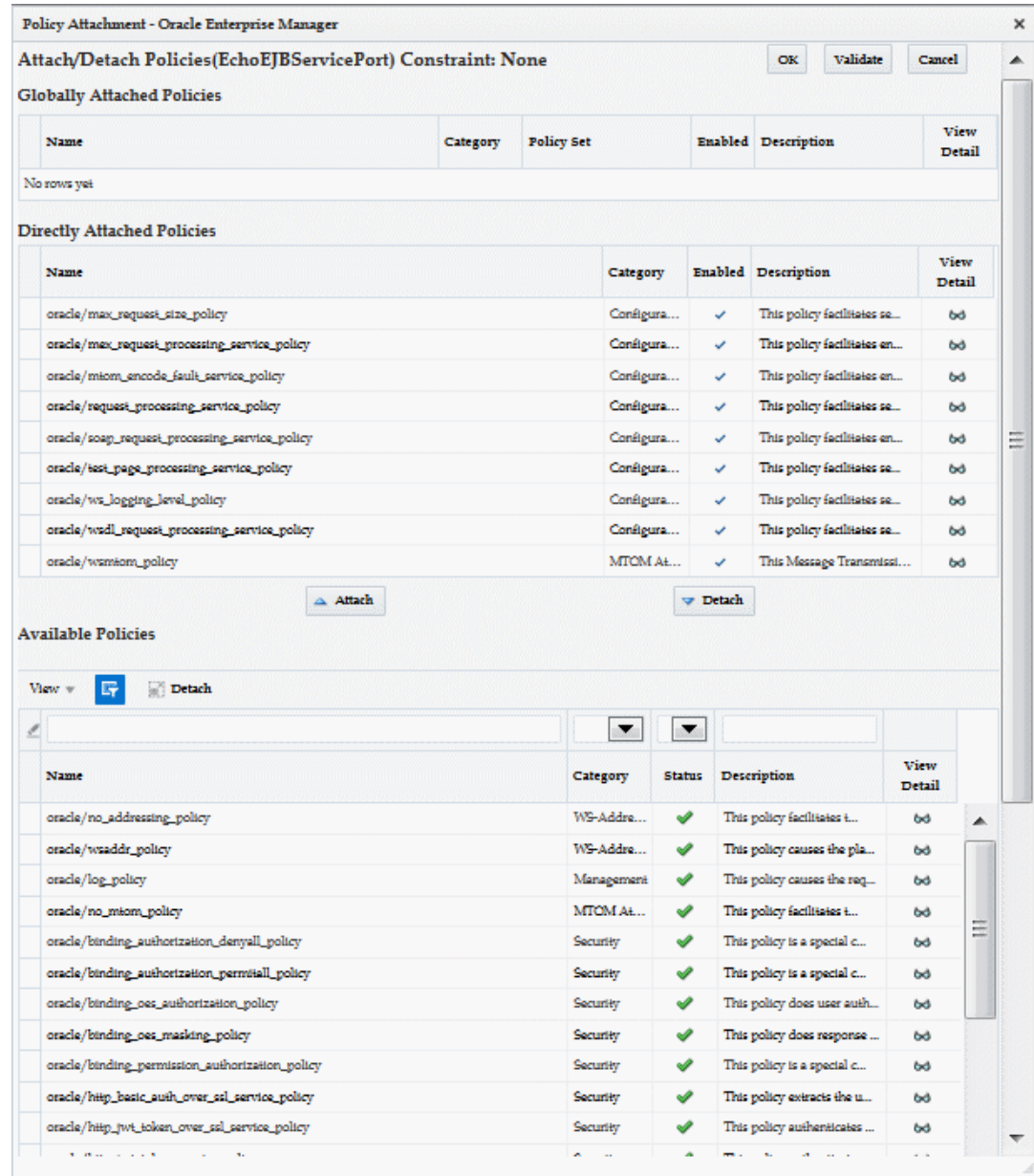
Figure 4-1 WSM Policy Subject Configuration Page with Directly Attached Policies

The screenshot displays the 'WSM Policy Subject Configuration' page. At the top, the user is logged in as 'weblogic' and the page title is 'jxwsejb30wGlobal'. The breadcrumb path is '/Domain_base_domain/base_domain/jxwsejb30wGlobal > Web Service Configuration > WSM Policy Subject Configuration'. The subject name is 'EchoEJBServicePort (SOAP Web Service)'. A 'Constraint' dropdown is set to 'None', and the 'Status' is 'Not Secure'. Below the configuration, there are two tables: 'Globally Attached Policies' (empty) and 'Directly Attached Policies'. The 'Directly Attached Policies' table has columns for 'Category/Policy Name', 'Effective', and 'Enabled'. The table lists several policies, with 'oracle/soap_request_processing_service_policy' highlighted in blue.

Category/Policy Name	Effective	Enabled
mtom		
oracle/wsmtom_policy	✓	✓
wsconfig		
oracle/mtom_encode_fault_service_policy	✓	✓
oracle/wsdl_request_processing_service_policy	✓	✓
oracle/soap_request_processing_service_policy	✓	✓
oracle/ws_logging_level_policy	✓	✓
oracle/test_page_processing_service_policy	✓	✓
oracle/mex_request_processing_service_policy	✓	✓
oracle/request_processing_service_policy	✓	✓
oracle/max_request_size_policy	✓	✓

3. Select an expression from the **Constraint** menu to optionally apply a constraint to the policy set that determines the context in which the policy set is relevant.
For more information about specifying a constraint, see "[Run-time Constraints in Policy Sets](#)".
4. Click the **Attach/Detach** option, located above the Category/Policy Name table.
The Policy Attachment dialog is displayed, as shown in [Figure 4-2](#).

Figure 4-2 Attaching Policies to a Web Service



- 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, as shown in [Figure 4-3](#). Click **OK** when you are finished reviewing the details of the policy.

Figure 4-3 Viewing Details about a Policy



6. Select a policy from the Available Policies list, and click **Attach**.
The selected policy moves to the Directly Attached Policies table.

Note:

The list of available policies is filtered based on whether you are attaching a policy to a SOAP or RESTful web service or client.

7. Continue selecting and attaching policies as required. When you are finished, click **Validate** to verify that the combination of policies selected is valid.
An Information pop-up should display with a "Validation is successful" message. Click **OK** to close the pop-up.
8. Click **OK** at the top of the Policy Attachment page.
9. The Directly Attached Policies table on the WSM Policy Subject Configuration page now displays the newly attached policy, based on Category.

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](#)", 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](#)".

4.3.1.2 Attaching Policies Directly to Web Service Clients Using Fusion Middleware Control

This section describes how to attach a policy to an Oracle Infrastructure web service client, including Java EE web service clients, SOA reference, ADF Data Control (DC), and asynchronous web service Callback clients.



Note:

You can attach policies directly to RESTful web service clients at design time only. For more information, see [Attaching Policies to RESTful Web Service Clients Using Feature Classes](#).

4.3.1.2.1 Attaching Policies to SOA References Using Fusion Middleware Control

The following procedures describe how to attach policies to SOA references.

To attach policies to a SOA reference:

1. Navigate to the Web Service Endpoint page, and view the SOA reference, as described in Viewing SOA References in Administering Web Services.
2. Click the **Attach/Detach Policies** link at the top of the page.

The WSL Policy Subject Configuration page is displayed and any policies that are already globally and directly attached to the endpoint are displayed.

3. Select a constraint, provided multiple selections are available. The effective policies that are dynamically calculated shown on this page might change based on the selected constraint.
4. Click **Attach/Detach**. The Policy Attachment dialog box is displayed.

5. Select a policy from the Available Policies list, and click **Attach**.

The selected policy moves to the Directly Attached Policies table.



Note:

The list of available policies is filtered based on whether you are attaching a policy to a SOAP or RESTful web service or client.

6. 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.
7. Click **Attach** when you are sure that you want to attach the policy or policies.
8. The Directly Attached Policies table on the WSM Policy Subject Configuration page now displays the newly attached policy, based on Category.
9. Click **OK**.
10. You can click Return on the top-right corner of the page after policy attachment to return to the Web Service motoring page.

4.3.1.2.2 Attaching Policies to Connection-Based Web Service Clients Using Fusion Middleware Control

The following procedure describes how to attach policies to a connection-based web service client such as an ADF DC web service client or 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 *Developing Fusion Web Applications with Oracle Application Development Framework*.

To attach policies to a connection-based web service client:

1. In the navigation pane, expand Application Deployments.
2. In the navigation pane, expand the target application deployment and select the application.
3. In the content pane, select **Application Deployment**, then **ADF**, then **Configure ADF Connections**.
4. On the **ADF Connections Configuration** page, select a row in the **Web Service Connections** list.
5. Click **Advanced Connection Configuration** and select the web service client from the list.
6. On the Web Service Client page, select the **WSM Policies** tab.
7. Click the **Attach/Detach Policies** link at the top of the page.

The WSM Policy Subject Configuration page is displayed and any policies that are already globally and directly attached to the endpoint are displayed.

8. Click the **Attach/Detach** option, located above the Category/Policy Name table to open the Policy Attachment dialog.
9. In 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.
10. Click **Attach** when you are sure that you want to attach the policy or policies.
11. Click **OK**.

4.3.1.2.3 Attaching Policies to Asynchronous Web Service Callback Clients Using Fusion Middleware Control

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 *Developing Oracle Infrastructure Web Services*.

To attach policies to an asynchronous Callback client:

1. Navigate to the Web Service Endpoint page, as described in "Viewing the Details for a Web Service Endpoint Using Fusion Middleware Control" in *Administering Web Services*.
2. From the **Attach/Detach Policies** drop down, select **Callback Client**.

The WSM Policy Subject Configuration page is displayed and any policies that are already globally and directly attached to the endpoint are displayed.

3. Click the **Attach/Detach Policies** link at the top of the page.

4. In the Directly Attached Policies section of the page, click **Attach/Detach**.
5. In the **Available Policies** section of the page, select one or more policies that you want to attach and click **Attach**.
6. Click **Validate** to validate if the combination of policies is valid.
7. Click **OK**.
8. You can click **Return** to return to the policy subject monitoring page.

4.3.1.2.4 Attaching Policies to Java EE Web Service Clients Using Fusion Middleware Control

OWSM policies are attached to a WebLogic Java EE web service client using Fusion Middleware control.

 **Note:**

For WebLogic Java EE web service client policy attachments:

- Only OWSM security policies can be attached.
- OWSM policies and WebLogic web service policies cannot be attached to the same client. If a Java EE client has any WebLogic policies attached, the WSM Policies tab is not displayed and you cannot attach OWSM security policies. Instead, the WebLogic Policy Violations tab is displayed showing violation details about the WebLogic web service policies attached to the client.

Note that WebLogic policies can be attached using the WebLogic Server Remote Console. You cannot attach WebLogic policies using Fusion Middleware Control.

- Oracle recommends that you attach OWSM policies to a web service client post-deployment. If you attach OWSM 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:

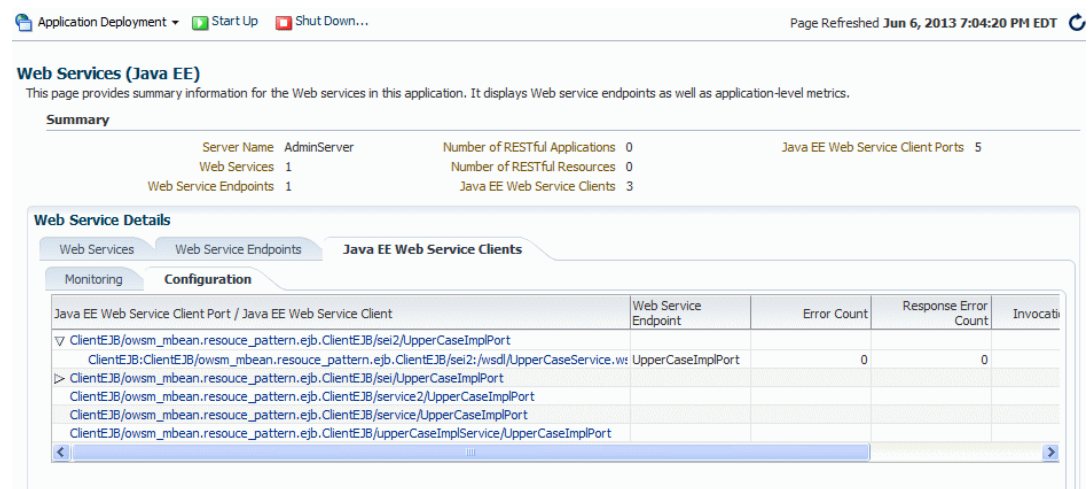
1. a. Navigate to the Java EE Web Service Client page, as described in "Viewing the Web Services Summary Page for an Application" in *Administering Web Services*.
2. Select the **Java EE Web Service Clients** link to navigate to WSM Policy Subject Configuration page for direct policy attachment.
3. Select the **Configuration** tab to view the available client ports to which you can attach policies, as shown in [Figure 4-4](#).

 **Note:**

If a port is associated with a run-time client instance, 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 "Roadmap for Developing JAX-WS Web Service Clients" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.

Figure 4-4 Java EE Web Service Clients



Application Deployment Start Up Shut Down... Page Refreshed Jun 6, 2013 7:04:20 PM EDT

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	Number of RESTful Applications	0	Java EE Web Service Client Ports	5
Web Services	1	Number of RESTful Resources	0		
Web Service Endpoints	1	Java EE Web Service Clients	3		

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	Invocati
ClientEJB/owsm_mbean.resouce_pattern.ejb.ClientEJB/sei2/UpperCaseImplPort				
ClientEJB:ClientEJB/owsm_mbean.resouce_pattern.ejb.ClientEJB/sei2:/wsdl/UpperCaseService.w	UpperCaseImplPort	0	0	
ClientEJB/owsm_mbean.resouce_pattern.ejb.ClientEJB/sei/UpperCaseImplPort				
ClientEJB/owsm_mbean.resouce_pattern.ejb.ClientEJB/service/UpperCaseImplPort				
ClientEJB/owsm_mbean.resouce_pattern.ejb.ClientEJB/service/UpperCaseImplPort				
ClientEJB/owsm_mbean.resouce_pattern.ejb.ClientEJB/UpperCaseImplService/UpperCaseImplPort				

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 4-5](#).

Figure 4-5 Attaching Policies to WebLogic Java EE Web Service Client Ports

The screenshot shows the configuration page for a Java EE Web Service Client Port. The breadcrumb path is: Web Services > Java EE Web Service Client Port > ClientEJB/owsm_mbean.resouce_ejb.ClientEJB/sei2/UpperCaseImplPort (Java EE Web Service Client Port). The page is titled "WSM Policies" and has two tabs: "Attach/Detach" and "Disable". Below the tabs is a table with the following data:

Name	Category	Policy Reference Status
oracle/wss10_username_token_with_message_protection_client_policy	Security	Enabled

Below the table is the "Security Configuration Details" section, which includes "Apply" and "Revert" buttons. It contains a table with the following data:

Name	Current Value	Original Value
reference.priority		
keystore.recipient.alias		orakey
csf-key		basic.credentials
keystore.sig.csf.key		
keystore.enc.csf.key		
ignore.timestamp.in.response		false
sc.token.lifetime		

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 WSM 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 "[Overriding Configuration Properties at the Web Service Client Application Level Using Fusion Middleware Control](#)".
8. You can click **Return** to return to the policy subject monitoring page.

4.3.1.3 Enabling or Disabling Directly Attached Policies Using Fusion Middleware Control

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.

To enable or disable a policy attached to an endpoint (port):

1. Navigate to the home page for the web service, as described in "Navigating to the Web Services Summary Page for an Application" in *Administering Web Services*.
2. In the **Web Service Details** section of the page, select the Web Service Endpoints tab to display a list of the web service endpoints in the application.
3. Click the name of an endpoint to navigate to the **Web Service Endpoint** page for a particular web service.
4. Click the **Attach/Detach Policies** link, located near the top of the page. The policies that are already globally and directly attached to the endpoint are displayed.
5. Select a policy from the **Directly Attached Policies** list, and click **Enable** or **Disable** to enable or disable the policy, respectively.

4.3.1.4 Detaching Directly Attached Policies Using Fusion Middleware Control

To detach a policy from a web service:

1. Navigate to the home page for the web service, as described in "Viewing the Web Services Summary Page for an Application" in *Administering Web Services*.
2. Select the **Web Service Endpoints** tab view the endpoints in the application and select the endpoint from which you want to detach the policies.
3. On the Web Service Endpoint page, select click the **Attach/Detach Policies** link, located near the top of the page. The policies that are globally and directly attached to the endpoint are displayed.
4. In the Directly Attached Policies section of the page, click **Attach/Detach**.
5. In the Directly Attached Policies table, select the policy to be detached, and click **Detach**.
6. Click **OK** to return to the Web Service Endpoint page.

4.3.2 About Attaching Policies Globally Using Fusion Middleware Control

Learn how to manage and create policy sets using Fusion Middleware Control.

 **Note:**

Global policy attachments are supported for RESTful web services and clients, Oracle Infrastructure web services and clients, and Java EE web services and clients.

For WebLogic Java EE web services:

- Only a subset of OWSM policies can be attached. For more information, see [OWSM Policies Supported for Java EE Web Services and Clients](#)
- OWSM policies and WebLogic web service policies cannot be attached to the same endpoint. If a Java EE endpoint has any WebLogic policies attached and you create a policy set that applies to that endpoint, the OWSM policies will be ignored when the effective policy for that endpoint is calculated.

Note that WebLogic policies can be attached using the WebLogic Server Remote Console. You cannot attach WebLogic policies using Fusion Middleware Control.

For RESTful web services:

- You can attach OWSM policies to RESTful web services and clients that are built using Jersey 2.x JAX-RS RI only. For more information about securing RESTful web services and clients built using Jersey 2.x JAX-RS RI, see "Securing RESTful Web Services and Clients" in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*.
- Only a subset of OWSM security policies can be attached. For more information, see [OWSM Policies Supported for RESTful Web Services and Clients](#)

For SOA composite services, only a subset of OWSM policies apply. For more information, see "[OWSM Policies Supported for SOA Composite Services and Clients](#)".

Policy sets provide a means to attach policies globally to a range of endpoints of the same type. For more information about global policy attachments, see "Understanding Global Policy Attachments Using Policy Sets" in *Understanding Oracle Web Services Manager*.

It includes the following topics:

- [Navigating to the WSM Policy Set Summary Page Using Fusion Middleware Control](#)
- [Viewing the Configuration of a Policy Set Using Fusion Middleware Control](#)
- [Creating a Policy Set Using Fusion Middleware Control](#)
- [Cloning a Policy Set Using Fusion Middleware Control](#)
- [Editing a Policy Set Using Fusion Middleware Control](#)
- [Specifying Run-time Constraints in a Policy Set Using Fusion Middleware Control](#)
- [Enabling and Disabling a Policy Set Using Fusion Middleware Control](#)
- [Deleting Policy Sets Using Fusion Middleware Control](#)

4.3.2.1 Navigating to the WSM Policy Set Summary Page Using Fusion Middleware Control

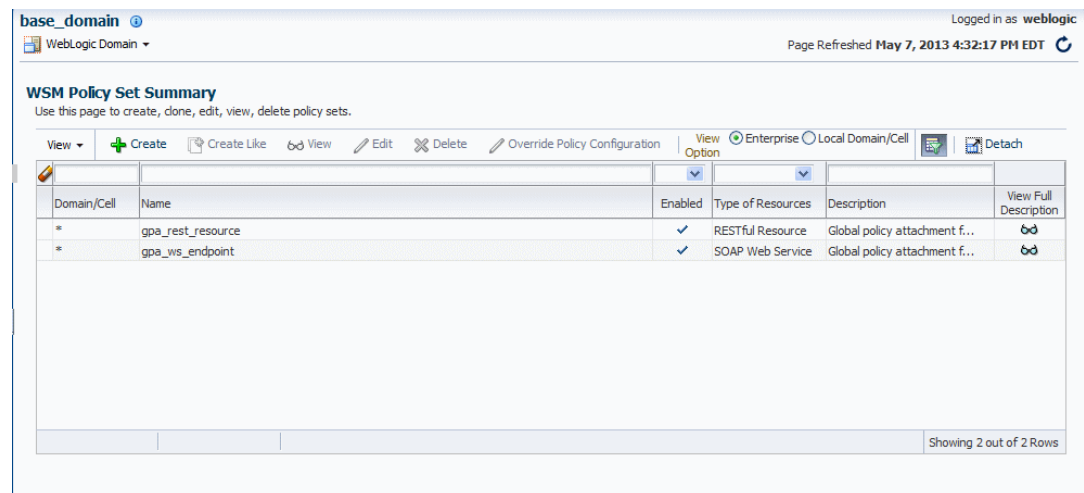
You can manage your policy sets at the domain level from the **WSM Policy Set Summary** page. From this page, you can see a list of all the existing policy sets or view the details of an individual policy set, create a new policy set, and copy, edit, or delete existing policy sets. You can also configure overrides for policy references in the policy sets.

To navigate to the **WSM 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 **WSM Policy Sets**.

The WSM Policy Set Summary page is displayed, as shown in [Figure 4-6](#).

Figure 4-6 WSM Policy Set Summary Page



4.3.2.2 Viewing the Configuration of a Policy Set Using Fusion Middleware Control

The following section describes how to view a policy set using Fusion Middleware Control.

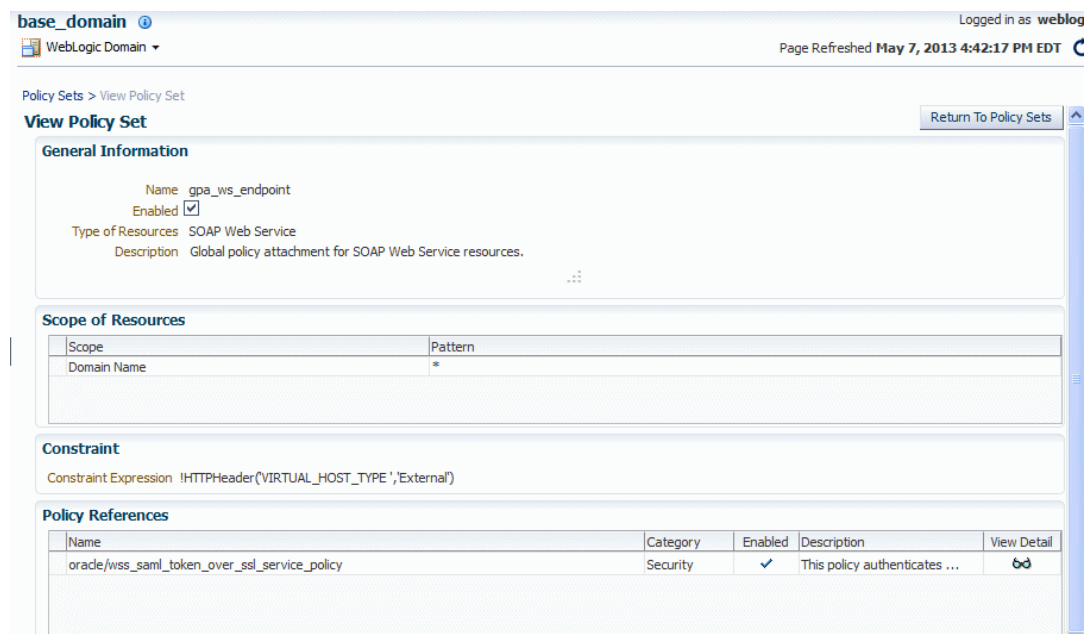
To view a policy set:

1. Navigate to the WSM Policy Set Summary page as described in "[Navigating to the WSM Policy Set Summary Page Using Fusion Middleware Control](#)".
2. In the WSM Policy Set Summary page, select a policy set from the table and click **View**.

From this page you can view general information about the policy set, the scope of resources to which it applies, any constraints that are configured, and the policy references that are contained in the policy set.

3. When you are done viewing the policy set, click **Return to Policy Sets**.

Figure 4-7 Viewing a Policy Set



4.3.2.3 Creating a Policy Set Using Fusion Middleware Control

The following section describes how to create a policy set using Fusion Middleware Control.

To create a policy set:

1. Navigate to the WSM Policy Set Summary page as described in "[Navigating to the WSM Policy Set Summary Page Using Fusion Middleware Control](#)".
2. From the WSM 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 4-8](#), enter a name for the policy set in the **Name** field.

Figure 4-8 Enter General Information Page

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 "Understanding Policy Subjects" in *Understanding Oracle Web Services Manager*.

 **Note:**

When creating policy sets, the SOAP Web Service and SOAP Web Service Client subject types refer both to Oracle Infrastructure web services and clients and to Java EE web services and clients. Non-security policies are ignored when the effective policy set for Java EE endpoints is calculated.

6. Optionally, add a description of the policy set in the **Description** field, 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 "Subject Types and Scope of Resources" in *Understanding Oracle Web Services Manager*.

 **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 SOAP Web Service, the resource scopes available are Domain, Application, Application Module or Connection, RESTful Application, Service, Web Service Endpoint, or Port.

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 Server Instance Name, the filtering conditions are ANDed together; for example, Domain("myDomain*") AND Server ("*SOA*"). For more information about specifying the resource type and scope, and an example that specifies multiple resource scopes, see "[About Defining the Type and Scope of Resources for Globally Attached Policies](#)".

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 "[Run-time Constraints in Policy Sets](#)".

10. In the Add Policy References page, select a policy from the Available Policies list, and click **Attach**.

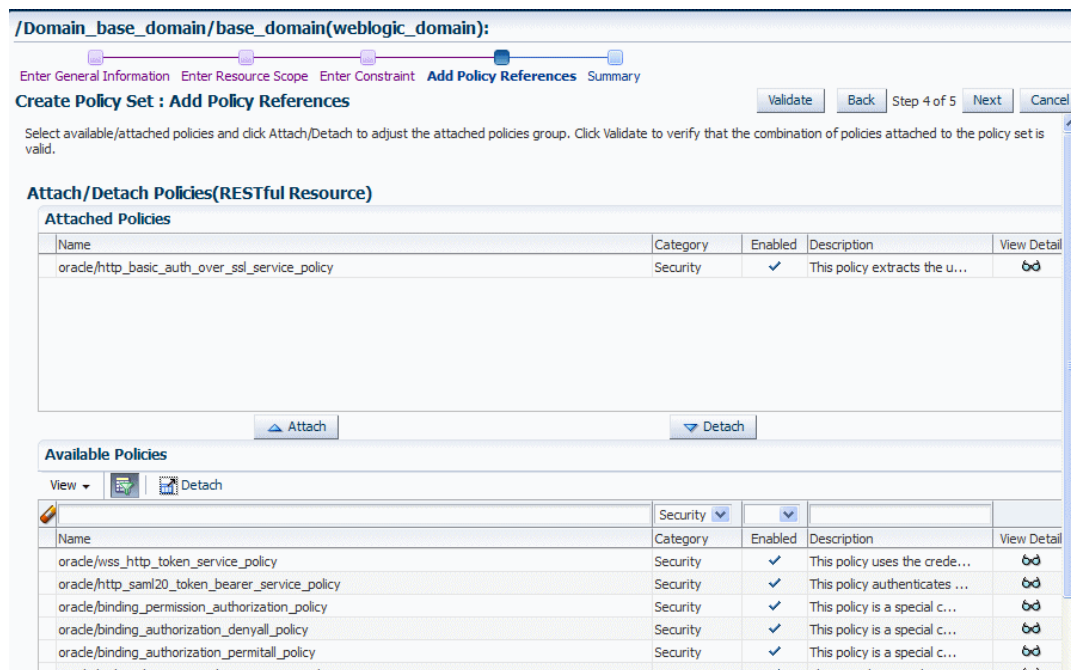
 **Note:**

You can use the query by example fields above the Available Policies table to filter the list of policies displayed. For example, to view only security policies, select **Security** from the drop-down menu above the Category column. To view only message protection policies, enter `message` in the field above the Name column and press **Enter**.

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.

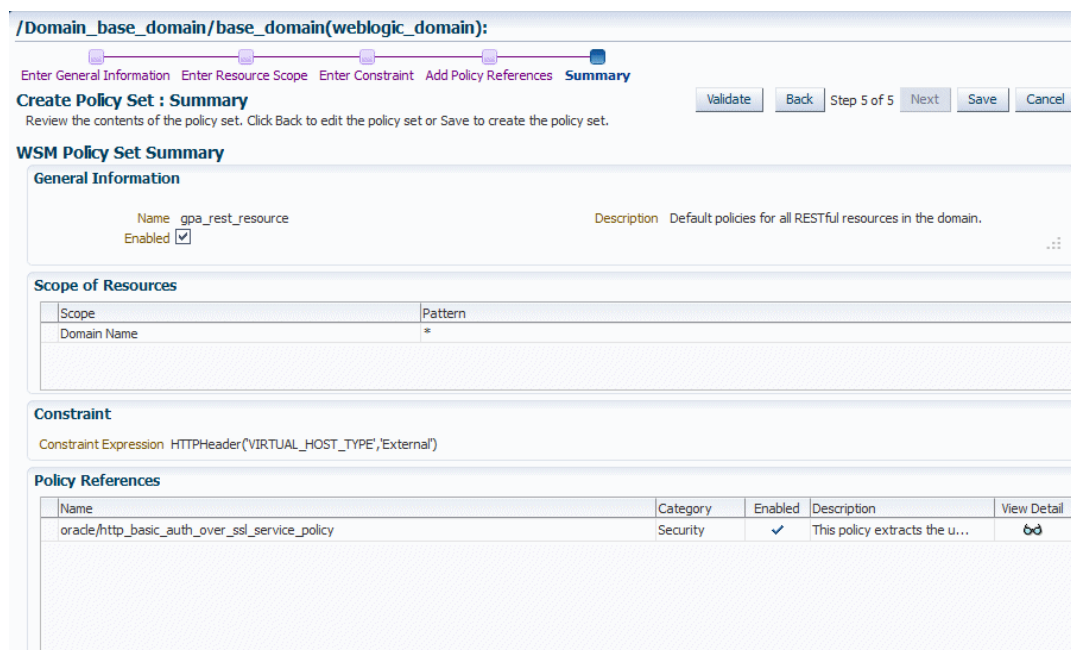
Figure 4-9 Add Policy References Page



12. Click **Next** to view the 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 4-10 Summary Page in Create Policy Set Wizard



4.3.2.4 Cloning a Policy Set Using Fusion Middleware Control

You can use an existing policy set as the base for a new policy set. The following section describes how to clone a new policy set from an existing policy set using Fusion Middleware Control.

Note that when you clone a new 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.

To clone a new policy set using an existing policy set:

1. Navigate to the WSM Policy Set Summary page as described in "[Navigating to the WSM Policy Set Summary Page Using Fusion Middleware Control](#)".
2. In the WSM 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 gpa_ws_endpoint, the name displayed for the copy is gpa_ws_endpoint_Copy.
 - The **Type of Resources** 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 "[Run-time Constraints in Policy Sets](#)".
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 Summary Page.
10. Review the policy set summary information. If you are satisfied with the policy set, click **Save**.

4.3.2.5 Editing a Policy Set Using Fusion Middleware Control

You can edit an existing policy set using Fusion Middleware Control by completing the following steps:

1. Navigate to the WSM Policy Set Summary page as described in "[Navigating to the WSM Policy Set Summary Page Using Fusion Middleware Control](#)".
2. In the WSM 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 "[Run-time Constraints in Policy Sets](#)".
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 Summary Page.
9. Review the policy set summary information. If you are satisfied with the policy set, click **Save**.

4.3.2.6 Specifying Run-time Constraints in a Policy Set 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. For more information about run-time constraints, see "[Run-time Constraints in Policy Sets](#)".

After you provide the general information about the policy set and specify the resource scope, 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 4-11](#). If you are creating a new policy set, these fields are blank.

Figure 4-11 Enter Constraint Page in Policy Set Wizard

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 4-11](#), 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')`.

4.3.2.7 Enabling and Disabling a Policy Set Using Fusion Middleware Control

The following section describes how to enable or disable a policy set 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 Using Fusion Middleware Control](#)". 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.

4.3.2.8 Deleting Policy Sets Using Fusion Middleware Control

To delete policy sets using Fusion Middleware Control:

1. Navigate to the WSM Policy Set Summary page as described in "[Navigating to the WSM Policy Set Summary Page Using Fusion Middleware Control](#)".
2. In the WSM 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**.

4.3.3 Viewing Policies Attached to a Web Service Using Fusion Middleware Control

Learn more about using the Fusion Middleware Control to view the OWSM policies attached to a web service.

To view the policies that are attached to a web service:

1. Navigate to the home page for the web service, as described in "Viewing the Web Services Summary Page for an Application" in *Administering Web Services*.
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 4-12](#) 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 Runtime Constraints in Policy Sets Using WLST](#)".

Because you can specify the priority of a globally or directly attached policy, as described in "[Specifying the Priority of a Policy Attachment](#)", 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](#)".

Figure 4-12 Policies Attached to an Oracle Infrastructure Web Service Endpoint

The screenshot shows the configuration page for the 'CalculatorPort (Web Service Endpoint)'. It includes a navigation breadcrumb 'Web Services > Web Service Endpoint', a help icon, and links for 'Web Services Test', 'Message Log', and 'Diagnostic Log'. A descriptive paragraph explains the page's purpose. Below this, there are two columns of configuration details: endpoint settings (Enabled, Asynchronous, Style, SOAP Version, Stateful, Implementation Type) and transport settings (Transport, Data Binding, Legacy Configuration, Implementation Class, WSDL Document). A tabbed interface shows 'OWSM Policies' selected. Underneath, there are two tables: 'Globally Attached Policies' and 'Directly Attached Policies'. The 'Directly Attached Policies' table includes an 'Attach/Detach' icon and columns for Policy Name, Category, Effective, Status, Total Violations, Authentication, Security Violations (Authorization, Confide), and Security Violations.

Policy Name	Category	Policy Set	Status	Total Violations	Authentication	Security Violations
orade/wss_username_token_service_pol	Security	/policysets/global/ws_1	Enabled	0	0	0

Policy Name	Category	Effective	Status	Total Violations	Authentication	Security Violations
orade/wss10_saml_hok_token_with_mes	Security	False	Enabled	0	0	0
orade/log_policy	Management	True	Enabled	0	n/a	n/a

Figure 4-13 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 4-13 Policies Attached to a WebLogic Java EE Web Service Endpoint

The screenshot shows the configuration page for the 'SimplePort (Web Service Endpoint)'. It includes a navigation breadcrumb 'Web Services > Web Service Endpoint', a help icon, and a link for 'Web Services Test'. Configuration details include Web Service Type (JAX-WS 2.1), Endpoint URI (/JAXWS_WEB/SimpleService), Transport (http), and WSDL Document (SimplePort). A tabbed interface shows 'OWSM Policies' selected. Below this, there is an 'Attach/Detach' icon and a table with columns for Policy Name, Category, Policy Reference Status, Total Violations, Authentication, and Security Violations (Authorization).

Policy Name	Category	Policy Reference Status	Total Violations	Authentication	Security Violations
orade/binding_authorization_deniall_...	Security	Enabled	0	0	0

4.3.4 Validating Policy Attachments

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

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.

4.3.5 About Validating a Policy Set

All OWSM policies attached to either a single policy subject or to multiple subjects (global attachment) adhere to the certain set rules.

In addition to validating that the policy set adheres to the rules described in "[Validating Policy Attachments](#)", 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](#)".

For troubleshooting information, see "[Overview of Policy Attachment Issues Using WLST](#)".

4.4 About Attaching Policies to Web Services and Clients Using WLST

You can attach OWSM policies to web services and clients using WLST.

The following topics describe how to attach policies using WLST, based on the web service or client type:

- [Viewing Available Policies Using WLST](#)
- [About Attaching Policies Directly to Java EE Web Services and Clients Using WLST](#)
- [About Attaching Policies Directly to RESTful and Oracle Infrastructure Web Services and Clients Using WLST](#)
- [About Attaching Policies Globally Using WLST](#)
- [Viewing Policies Attached to a Web Service with WLST](#)
- [Displaying the Effective Policy Set Using WLST](#)

4.4.1 Viewing Available Policies Using WLST

Use the WLST command to view the available policies.

To display a list of the available policies using WLST:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
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 : 115
security : oracle/http_basic_auth_over_ssl_service_policy
wsm : oracle/wsm10_policy
security : oracle/wss_username_token_client_policy
security : oracle/binding_authorization_denyall_policy
security : oracle/wss11_username_token_with_message_protection_service_policy
security : oracle/no_messageprotection_client_policy
security : oracle/wss_saml_token_over_ssl_service_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 : 55
security : oracle/http_basic_auth_over_ssl_service_policy
security : oracle/binding_authorization_denyall_policy
security : oracle/wss11_username_token_with_message_protection_service_policy
security : oracle/wss_saml_token_over_ssl_service_policy
security : oracle/wss10_saml_token_service_policy
security : oracle/binding_permission_authorization_policy
security : oracle/no_messageprotection_service_policy
security : oracle/wss10_x509_token_with_message_protection_service_policy
security : oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy
security : oracle/wss_saml_token_bearer_over_ssl_service_policy
security : oracle/wss_saml20_token_bearer_over_ssl_service_policy
security : oracle/wss_saml_bearer_or_username_token_service_policy
security : oracle/wss10_message_protection_service_policy
...
```

4.4.2 About Attaching Policies Directly to Java EE Web Services and Clients Using WLST

Policies can be attached directly to Java EE web services and clients using WLST.

 **Note:**

A web service cannot contain both a WebLogic web service policy and an OWSM web service policy. If you have a web service with a WebLogic web service policy, you must first detach it before attaching the OWSM web service policy.

Topics

- ["Viewing the Policies That Are Attached to a Java EE Web Service"](#)
- ["Viewing the Policies That Are Attached to a Java EE Web Service Client"](#)
- ["Attaching Policies Directly to a Java EE Web Service Using WLST"](#)
- ["Attaching Policies Directly to Java EE Web Service Clients Using WLST"](#)
- ["Detaching Directly Attached Policies from Java EE Web Service Clients Using WLST"](#)
- ["Enabling and Disabling Web Service Client Policies Using WLST"](#)

4.4.2.1 Viewing the Policies That Are Attached to a Java EE Web Service

Use the following procedure to view the policies that are attached to a web service client:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
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 Using WLST" in *Administering Web Services*.
3. Use the `listWebServicePorts` command to display the port name for a web service.

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

For example, to display the port for the `SimpleImplService` web service:

```
wls:/wls-domain/serverConfig> listWebServicePorts('/base_domain/AdminServer/
SimpleJAXWS','SimpleJAXWS#1!SimpleImplService',
'wls','SimpleImplService')
```

```
SimplePort
```

4. Use the `listWebServicePolicies` command to view the policies that are attached to a web service port.

```
listWebServicePolicies(application,moduleOrCompName,moduleType,serviceName,subjectName)
```

For example, to view the policies attached to the port `SimplePort` port and any policy override settings:

```
wls:/wls-domain/serverConfig> listWebServicePolicies('/base_domain/AdminServer/
SimpleJAXWS','SimpleJAXWS#1!SimpleImplService','wls',
'SimpleImplService','SimplePort')
```

```
SimplePort :
```

```
URI="oracle/wss_username_token_service_policy", category=security,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
```

The policy subject is secure in this context.

4.4.2.2 Viewing the Policies That Are Attached to a Java EE Web Service Client

Use the following procedure to view the policies that are attached to a web service client:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Use the `listWebServicesClients` WLST command to display a list of the web service clients in your application as described in "Viewing Web Service Clients Using WLST" in *Administering Web Services*.
3. Use the `listWebServiceClientPorts` command to display the port name for a web service.

```
listWebServiceClientPorts(application,moduleOrCompName,moduleType,serviceRefName)
```

For example, to display the port for the `service2` client:

```
wls:/wls-domain/serverConfig> listWebServiceClientPorts('/base_domain/AdminServer/  
ClientEJB','owsm_mbean.resource_pattern.ejb.ClientEJB/service2',  
'wls','owsm_mbean.resource_pattern.ejb.ClientEJB/service2')
```

```
UpperCaseImplPort
```

4. Use the `listWebServiceClientPolicies` command to view the policies that are attached to a web service port.

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

For example, to view the policies attached to the port `UpperCaseImplPort` port and any policy override settings:

```
wls:/wls-domain/serverConfig> listWebServiceClientPolicies('/base_domain/AdminServer/  
ClientEJB','owsm_mbean.resouce_pattern.ejb.ClientEJB/service2',  
'wls','owsm_mbean.resouce_pattern.ejb.ClientEJB/service2','UpperCaseImplPort')
```

```
UpperCaseImplPort :  
    URI="oracle/wss_username_token_client_policy", category=security,  
    policy-status=enabled; source=local policy set; reference-status=enabled;  
    effective=true
```

The policy subject is secure in this context.

4.4.2.3 Attaching Policies Directly to a Java EE 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. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. View the list of policies currently attached to the port as described in "[Viewing the Policies That Are Attached to a Java EE Web Service](#)".
3. View the list of available policies as described in "[Viewing Available Policies Using WLST](#)".
4. 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 `wss10_message_protection_service_policy` to the `SimplePort` of the `SimpleImplService` web service, use the following command:

```
wls:wls-domain/serverConfig> attachWebServicePolicy('/base_domain/AdminServer/
SimpleJAXWS', 'SimpleJAXWS#1!
SimpleImplService', 'wls', 'SimpleImplService', 'SimplePort', 'oracle/
wss10_message_protection_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/binding_authorization_denyall_policy` to the `SimplePort` of the `SimpleImplService`, use the following command:

```
wls:wls-domain/ServerConfig>attachWebServicePolicies ('/base_domain/
AdminServer/SimpleJAXWS', 'SimpleJAXWS#1!
SimpleImplService', 'wls', 'SimpleImplService', 'SimplePort', ["oracle/
binding_authorization_denyall_policy", "oracle/
wss_username_token_service_policy"])
```

Note:

The `policyURIs` are validated through the OWSM 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 Attachments](#)".

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

4.4.2.4 Attaching Policies Directly to Java EE Web Service Clients Using WLST

The following procedure describes how to attach policies to SOA references, Java EE web service clients.

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.

2. View the web service clients as described in "Viewing Web Service Clients Using WLST" in *Administering Web Services*.
3. Use the `listWebServiceClientPorts` command to display the port name for a web service client.

```
listWebServiceClientPorts(application,moduleOrCompName,moduleType,serviceRefName)
```

For example, to display the port for the client `service2`:

```
wls:/wls-domain/serverConfig> listWebServiceClientPorts('/base_domain/AdminServer/ClientEJB','owsm_mbean.resource_pattern.ejb.ClientEJB/service2',
'wls','owsm_mbean.resource_pattern.ejb.ClientEJB/service2')
```

```
UpperCaseImplPort
```

4. View the list of available policies as described in "[Viewing Available Policies Using WLST](#)".

To view only available client policies, set the `subject` argument to `client`. For example:

```
listAvailableWebServicePolicies("", "client")
```

5. To attach policies, do one of the following:

- Use the `attachWebServiceClientPolicy` command to attach a single policy to a Java EE web service client port.

```
attachWebServiceClientPolicy(application, moduleOrCompName, moduleType,
serviceRefName, portInfoName, policyURI, [subjectType=None])
```

Set the arguments as follows:

- Specify the name of the client application using the `application` argument.
- Specify the Web module name for the `moduleOrCompName` argument.
- Specify `wls` for the `moduleType` argument.
- Specify the name of the endpoint 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 `UpperCaseImplPort` of the `service2` client, use the following command:

```
wls:/wls_domain/serverConfig>attachWebServiceClientPolicy('/base_domain/
AdminServer/ClientEJB','owsm_mbean.resource_pattern.ejb.ClientEJB/service2',
'wls','owsm_mbean.resource_pattern.ejb.ClientEJB/
service2','UpperCaseImplPort','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/wss10_message_protection_client_policy` to the `UpperCaseImplPort` of the `service2` service, use the following command:

```
wls:/wls_domain/serverConfig> attachWebServiceClientPolicies('/base_domain/  
AdminServer/ClientEJB', 'owsm_mbean.resource_pattern.ejb.ClientEJB/service2',  
'wls', 'owsm_mbean.resource_pattern.ejb.ClientEJB/service2', 'UpperCaseImplPort',  
['oracle/wss_username_token_client_policy', 'oracle/  
wss10_message_protection_client_policy'])
```

 **Note:**

The policyURIs are validated through the OWSM 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 Attachments](#)".

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

4.4.2.5 Detaching Directly Attached Policies from Java EE Web Service and Clients Using WLST

The following procedure describes how to detach policies from a Java EE web service or client:

- "[Detaching Directly Attached Policies from a Java EE Web Service Using WLST](#)"
- "[Detaching Directly Attached Policies from Java EE Web Service Clients Using WLST](#)"

4.4.2.5.1 Detaching Directly Attached Policies from a Java EE Web Service Using WLST

To detach directly attached policies from a Java EE web service:

1. View the policies attached to the web service, as described in "[Viewing Policies Attached to a Web Service with WLST](#)"
2. To detach policies from a Java EE web service endpoint, do one of the following:
 - Use the `detachWebServicePolicy` command to detach a single policy from a web service endpoint.

```
detachWebServicePolicy(application, moduleOrCompName, moduleType,  
serviceName, subjectName, policyURI, [subjectType=None])
```

Set the arguments as follows:

- Specify the name of the application using the `application` argument.
- Specify the Web module name for the `moduleOrCompName` argument.
- Specify `wls` for the `moduleType` argument.
- Specify the name of the endpoint using the `subjectName` argument.
- Specify the policy to be detached 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 detach the policy `wss10_message_protection_service_policy` from the `SimplePort` of the `SimpleImplService` web service, use the following command:

```
wls:/wls_domain/serverConfig>detachWebServicePolicy('/base_domain/AdminServer/
SimpleJAXWS','SimpleJAXWS#1!
SimpleImplService','wls','SimpleImplService','SimplePort','oracle/
wss10_message_protection_service_policy')
```

- Use the `detachWebServicePolicies` command to detach multiple policies from a Java EE web service endpoint. You specify the multiple 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/binding_authorization_denyall_policy` from the `SimplePort` of the `SimpleImplService`, use the following command:

```
wls:/wls_domain/serverConfig> detachWebServicePolicies('/base_domain/AdminServer/
SimpleJAXWS','SimpleJAXWS#1!SimpleImplService',
'wls','SimpleImplService','SimplePort',['oracle/
binding_authorization_denyall_policy','oracle/
wss_username_token_service_policy'])
```

4.4.2.5.2 Detaching Directly Attached Policies from Java EE Web Service Clients Using WLST

To detach policies from a Java EE web service client, do one of the following:

1. View the policies attached to the web service client, as described in "[Viewing the Policies That Are Attached to a Java EE Web Service](#)".
2. Do one of the following:
 - Use the `detachWebServiceClientPolicy` command to detach a single policy from a Java EE web service client port. Specify the policy to be detached using the `policyURI` argument.

```
detachWebServiceClientPolicy(application, moduleOrCompName, moduleType,
serviceRefName, portInfoName, policyURI, [subjectType=None])
```

For example, to detach the client policy `oracle/wss_username_token_client_policy` from the `UpperCaseImplPort` of the `service2` service, use the following command:

```
wls:/wls_domain/serverConfig> detachWebServiceClientPolicy('/base_domain/
AdminServer/ClientEJB','owsm_mbean.resource_pattern.ejb.ClientEJB/service2',
'wls','owsm_mbean.resource_pattern.ejb.ClientEJB/
service2','UpperCaseImplPort','oracle/wss_username_token_client_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_client_policy` and `oracle/wss10_message_protection_client_policy` from the `UpperCaseImplPort` of the `service2` service, use the following command

```
wls:/wls_domain/serverConfig> detachWebServiceClientPolicies('/base_domain/
AdminServer/ClientEJB','owsm_mbean.resouce_pattern.ejb.ClientEJB/service2',
'wls','owsm_mbean.resouce_pattern.ejb.ClientEJB/service2','UpperCaseImplPort',
```

```
['oracle/wss_username_token_client_policy','oracle/
wss10_message_protection_client_policy']])
```

4.4.2.6 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 "Viewing Web Service Clients Using WLST" in *Administering Web Services*.
2. Use the `listWebServiceClientPorts` command to display the port name and endpoint URL for a web service client, as described in "[Attaching Policies Directly to Java EE Web Service Clients Using WLST](#)".
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 `UpperCaseImplPort` of the Web module `WssUsernameClient`.

```
wls:/wls-domain/serverConfig>enableWebServiceClientPolicy('/base_domain/AdminServer/
ClientEJB','owsm_mbean.resouce_pattern.ejb.ClientEJB/service2',
'wls','owsm_mbean.resouce_pattern.ejb.ClientEJB/
service2','UpperCaseImplPort','oracle/wss_username_token_client_policy',true)
```

4.4.3 About Attaching Policies Directly to RESTful and Oracle Infrastructure Web Services and Clients Using WLST

You can attach and manage web service policies and policy sets using WLST.

Policies can be attached either directly or globally to RESTful and Oracle Infrastructure web services using WLST, as described in the following sections:

- [Identifying and Selecting the Policy Subject Using WLST](#)
- [Attaching Policies Directly Using WLST](#)
- [Enabling and Disabling Directly Attached Policies Using WLST](#)
- [About Detaching Directly Attached Policies Using WLST](#)

Note:

You can attach OWSM policies to RESTful web services and clients that are built using Jersey 2.x JAX-RS RI only. For more information about securing RESTful web services and clients built using Jersey 2.x JAX-RS RI, see "Securing RESTful Web Services and Clients" in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*.

4.4.3.1 Identifying and Selecting the Policy Subject Using WLST

You can view the policy subjects for a web service or a SOA web service directly by using the `listWSMPolicySubjects()` command. The `listWSMPolicySubjects()` command displays endpoint information, such as the application, assembly, and subject patterns for the web service.

You can navigate to the policy subject using the `selectWSMPolicySubject` command. You must start a session using `beginWSSession` before performing any policy management edits.

Example of identifying policy subjects:

To simplify searching for a particular subject, the `application`, `assembly`, or `subject` argument can specify a pattern containing the wildcard character (*). In this case, all the subjects matching that pattern will be listed. For example, invoking the `listWSMPolicySubjects` command with ('jax*') as the argument returns all subjects belonging to the `jaxrs_pack1` and `jaxwsejb30ws` applications:

```
wls:/base_domain/serverConfig> listWSMPolicySubjects('jax*')
```

```
Application: /weblogic/base_domain/jaxrs_pack1
```

```
Assembly: #jaxrs_pack1.war
```

```
Subject: REST-Resource(Jersey)
```

```
Application: /weblogic/base_domain/jaxwsejb30ws
```

```
Assembly: #jaxwsejb
```

```
Subject: WS-Service({http://ejb.oracle.com/
targetNamespace}EchoEJBService#EchoEJBServicePort)
```

```
Subject: WS-Service({http://www.oracle.com/jaxws/tests/
concrete}WsdConcreteService#WsdConcretePort)
```

```
Subject: WS-Service({http://www.oracle.com/jaxws/
tests}CalculatorService#CalculatorPort)
```

Examples of selecting policy subjects:

You can use the information provided to build the `selectWSMPolicySubject` command:

```
wls:/base_domain/serverConfig> selectWSMPolicySubject
('jaxwsejb30ws', '#jaxwsejb', 'WS-SERVICE({http://ejb.oracle.com/
targetNamespace}EchoEJBService#EchoEJBServicePort)')
```

The policy subject is selected for modification.

As an alternative, you can use the `selectWSMPolicySubject` command to navigate to the policy subject. The following example assumes that you already know part of the application name:

```
wls:/base_domain/serverConfig> selectWSMPolicySubject ('*ejb30ws') jaxwsejb30wsSelect
any of the application name to proceed.wls:/base_domain/serverConfig>
selectWSMPolicySubject('jaxwsejb30ws')
```

```
#jaxwsejb
```

Select any of the assembly name to proceed.

```
wls:/base_domain/serverConfig> selectWSMPolicySubject(assembly='#jaxwsejb')

WS-Service({http://example.com/jaxws/tests/concrete}WsdConcreteService#WsdConcretePort)
WS-Service({http://
example.com/}JaxwsWithHandlerChainBeanService#JaxwsWithHandlerChainBeanPort)
WS-Service({http://soapinterop.org/
DoclitWrapperWTJ}DoclitWrapperWTJService#DoclitWrapperWTJPort)
WS-Service({http://example.com/jaxws/tests}CalculatorService#CalculatorPort)
WS-Service({http://ejb.example.com/targetNamespace}EchoEJBService#EchoEJBServicePort)

Select any of the subject name to proceed.

wls:/base_domain/serverConfig> selectWSMPolicySubject (subject='WS-Service({http://
www.oracle.com/jaxws/tests/concrete}WsdConcreteService#WsdConcretePort)')

The policy subject is selected for modification.
```

4.4.3.2 Attaching Policies Directly Using WLST

Note:

You can attach OWSM policies to RESTful web services and clients that are built using Jersey 2.x JAX-RS RI only. For more information about securing RESTful web services and clients built using Jersey 2.x JAX-RS RI, see "Securing RESTful Web Services and Clients" in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*.

For SOA composite services and clients, only a subset of OWSM policies apply. For more information, see "[OWSM Policies Supported for SOA Composite Services and Clients](#)".

When attaching policies directly to subjects within a clustered server environment, the policy attachment details will be propagated to other servers in the cluster following a brief delay. To expedite the propagation of the information, perform one of the following steps:

- Restart the other servers in the cluster.
- Configure the cache refresh properties to minimize the delay. For more information, see "[High Availability Configuration and Cache Management Using Fusion Middleware Control](#)".

Policies can be attached to a policy subject, such as a web service or a web service client. The following sections describe how to attach policies to a single web service port and to web service clients.

Use the following procedure to attach a single policy, or multiple policies, to a single web service or client endpoint using WLST.

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. View the list of policies currently attached to the port as described in "[Viewing Policies Attached to a Web Service with WLST](#)".
3. View the list of available policies as described in "[Viewing Available Policies Using WLST](#)".
4. Begin a session using the `beginWSMSession` command, for example:

```
wls:/wls_domain/serverConfig> beginWSMSession()
```

Session started for modification.

5. Identify and select the policy subject that you want to work with. See ["Identifying and Selecting the Policy Subject Using WLST"](#).
6. To attach policies, do one of the following:

- Use the `attachWSMPolicy` command to attach a single policy to a web service port. Specify the policy to be attached using the `uri` argument. If you specify a policy that is already attached or exists, then this command enables the policy if it is disabled.

```
attachWSMPolicy(uri)
```

For example, to attach the policy `oracle/wss_username_token_service_policy` to the `WsdConcretePort` of the `WsdConcreteService`, use the following command:

```
wls:/wls_domain/serverConfig> attachWSMPolicy("oracle/  
wss_username_token_service_policy")
```

Policy reference "oracle/wss_username_token_service_policy" added.

- Use the `attachWSMPolicies` 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.

```
attachWSMPolicies(policyURIs, [subjectType=None])
```

For example, to attach the policies `oracle/wss_username_token_service_policy` and `oracle/wsrml0_policy` to the `WsdConcretePort` of the `WsdConcreteService`, use the following command:

```
wls:/wls_domain/serverConfig> attachWSMPolicies(["oracle/  
wss_username_token_service_policy","oracle/wsrml0_policy"])
```

Policy reference "oracle/wss_username_token_service_policy" added. Policy reference "oracle/wsrml0_policy" added.

Note:

The policy URIs are validated through the OWSM 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 Attachments"](#).

7. Optionally, specify configuration override properties for the policy using the `setWSMPolicyOverride` command. For example:

```
wls:/wls_domain/serverConfig> setWSMPolicyOverride("oracle/  
wss_username_token_service_policy", "reference.priority", "10")
```

The configuration override property "reference.priority" having value "10" has been added to the reference to policy with URI "oracle/wss_username_token_service_policy".

- Commit the session using the `commitWSSession` command, for example:

```
wls:/wls_domain/serverConfig> commitWSSession()

The policy set for subject "/weblogic/base_domain/jaxwsejb30ws|#jaxwsejb|WS-
Service({http://ejb.oracle.com/targetNamespace}EchoEJBService#EchoEJBServicePort)"
was saved successfully.
```

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

4.4.3.3 Enabling and Disabling Directly Attached Policies Using WLST

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.

To enable or disable a policy or multiple policies attached to an endpoint (port):

- Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
- 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 `WsdLConcreteServicePort`, use the following command. The results indicate that there are no policies currently attached to the port

```
wls:/base_domain/serverConfig> listWebServicePolicies('/base_domain/AdminServer/
jaxwsejb30ws','jaxwsejb','web','WsdLConcreteService','WsdLConcretePort')

WsdLConcretePort :
    URI="oracle/mex_request_processing_service_policy",
category=wsconfig, policy-status=enabled; source=local policy set;
reference-status=enabled; effective=true
    URI="oracle/mtom_encode_fault_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set;
reference-status=enabled; effective=true
    URI="oracle/max_request_size_policy", category=wsconfig, policy-
status=enabled; source=local policy set;
reference-status=enabled; effective=true
    Property name="max.request.size", value="-1"
    URI="oracle/request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set;
reference-status=enabled; effective=true
    URI="oracle/soap_request_processing_service_policy",
category=wsconfig, policy-status=enabled; source=local policy set;
reference-status=enabled; effective=true
    URI="oracle/ws_logging_level_policy", category=wsconfig, policy-
status=enabled; source=local policy set;
reference-status=enabled; effective=true
    Property name="logging.level", value=""
    URI="oracle/test_page_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set;
reference-status=enabled; effective=true
    URI="oracle/wsdL_request_processing_service_policy",
category=wsconfig, policy-status=enabled; source=local policy set;
reference-status=enabled; effective=true
```

```

        URI="oracle/
wss10_saml20_token_with_message_protection_service_policy", category=security,
policy-status=enabled; source=local policy set;
reference-status=enabled; effective=true
        URI="oracle/binding_authorization_denyall_policy",
category=security, policy-status=enabled; source=local policy set;
reference-status=enabled; effective=true

```

The policy subject is secure in this context.

3. Begin a session using the `beginWSMSession` command, for example:

```
wls:/wls_domain/serverConfig> beginWSMSession()
```

Session started for modification.

4. Select the policy subject you want to work with. See "[Identifying and Selecting the Policy Subject Using WLST](#)".
5. Enable or disable a single policy using the `enableWSMPolicy` command and setting the `enable` argument to `true` or `false`, respectively.

```
enableWSMPolicy(policyURI,[enable], [subjectType=None] )
```

For example, to disable the `oracle/wss10_saml20_token_with_message_protection_service_policy`, enter the following command:

```
wls:/base_domain/domainRuntime> enableWSMPolicy('oracle/
wss10_saml20_token_with_message_protection_service_policy',false)
```

6. Enable or disable multiple policies attached to a port using the `enableWSMPolicies` command and setting the `enable` argument to `true` or `false`, respectively.

```
enableWSMPolicies(policyURIs,[enable],[subjectType=None] )
```

For example:

```
wls:/base_domain/domainRuntime> enableWSMPolicies(['oracle/
binding_authorization_denyall_policy', 'oracle/
wss10_saml20_token_with_message_protection_service_policy'],true)
```

7. Commit the session using the `commitWSMSession` command, for example:

```
wls:/wls_domain/serverConfig> commitWSMSession()
```

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

4.4.3.4 About Detaching Directly Attached Policies Using WLST

Policies can be detached from a policy subject, such as a web service endpoint or from a client endpoint using WLST, as described in the following sections:

You can find details about detaching policies in the following sections:

- [Detaching Policies from a Service Endpoint](#)
- [Detaching Policies from a Client Endpoint](#)

4.4.3.4.1 Detaching Policies from a Service Endpoint

To detach policies, first start a session (`beginWSSession`), and then select the endpoint from which you want to detach policies. See "[Identifying and Selecting the Policy Subject Using WLST](#)".

Then do one of the following:

- Use the `detachWSMPolicy` command to detach a single policy from a web service endpoint. Specify the policy to be detached using the `policyURI` argument.

```
detachWSMPolicy(policyURI, [subjectType=None])
```

For example, to detach the policy `oracle/binding_authorization_denyall_policy` from the selected subject:

```
wls:/wls_domain/serverConfig> detachWSMPolicy("oracle/  
binding_authorization_denyall_policy")
```

- Use the `detachWSMPolicies` command to detach multiple policies from a web service endpoint. Specify the policies to be detached using the `policyURIs` argument.

```
detachWSMPolicies(policyURIs, [subjectType=None])
```

For example, to detach the policies `oracle/wss_username_token_service_policy` and `oracle/wss10_message_protection_service_policy` from the selected subject, use the following command:

```
wls:/wls_domain/serverConfig> detachWSMPolicies(["oracle/  
wss_username_token_service_policy","oracle/wss10_message_protection_service_policy"])
```

4.4.3.4.2 Detaching Policies from a Client Endpoint

To detach policies, first start a session (`beginWSSession`), and then select the endpoint from which you want to detach policies. See "[Identifying and Selecting the Policy Subject Using WLST](#)".

Then do one of the following:

- Use the `detachWSMPolicy` command to detach a single policy from a web service client port.

```
detachWSMPolicy(policyURI, [subjectType=None])
```

For example, to detach the client policy `oracle/wss_username_token_client_policy` from selected subject, use the following command:

```
wls:/wls_domain/serverConfig> detachWSMPolicy("oracle/  
wss_username_token_client_policy")
```

```
Policy reference "oracle/wss_username_token_client_policy" removed.
```

- Use the `detachWSMPolicies` command to detach multiple policies from a web service client port. You specify multiple policies to be detached using the `policyURIs` argument.

```
detachWSMPolicies(policyURIs, [subjectType=None])
```

For example, to detach the policies `oracle/wss_username_token_client_policy` and `oracle/wss11_message_protection_client_policy` from the selected client subject, use the following command:

```
wls:/wls_domain/serverConfig> detachWSMPolicies(["oracle/  
wss_username_token_client_policy","oracle/wss11_message_protection_client_policy"])  
  
Policy reference "oracle/wss_username_token_client_policy" removed.  
Policy reference "oracle/wss11_message_protection_client_policy" removed.
```

 **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.

4.4.4 About Attaching Policies Globally Using WLST

You can use WLST commands to create policy sets and manage global policy attachments.

 **Note:**

Global policy attachments using policy sets are supported for Oracle Infrastructure web services and clients, Java EE web services and clients, and RESTful web services and clients. However, if a policy set includes non-security policies, those non-security policies are ignored and therefore not included in the effective policy sets calculated for Java EE web services and clients.

Globally attached policies are not supported for standalone Java EE clients.

You can attach OWSM policies to RESTful web services and clients that are built using Jersey 2.x JAX-RS RI only. For more information about securing RESTful web services and clients built using Jersey 2.x JAX-RS RI, see "Securing RESTful Web Services and Clients" in *Developing and Securing RESTful Web Services for Oracle WebLogic Server*.

For SOA composite services and clients, only a subset of OWSM policies apply. For more information, see "[OWSM Policies Supported for SOA Composite Services and Clients](#)".

For Java EE web services and clients, only a subset of OWSM policies are supported. For more information, see [OWSM Policies Supported for Java EE Web Services and Clients](#)

These tasks are described in the following sections:

- [Viewing a List of Policy Sets](#)
- [Displaying the Configuration of a Policy Set](#)
- [Managing Sessions Using WLST](#)
- [Creating a New Policy Set Using WLST](#)
- [Cloning a Policy Set using WLST](#)
- [Editing a Policy Set](#)

- [Validating a Policy Set](#)
- [Enabling and Disabling a Policy Set](#)
- [Disabling a Globally Attached Policy](#)
- [Deleting Policy Sets Using WLST](#)
- [Specifying Runtime Constraints in Policy Sets Using WLST](#)

 **Note:**

To view the help for the WLST commands described in this section, connect to a running instance of the server and enter `help('wsmManage')`.

4.4.4.1 Viewing a List of Policy Sets

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" in *Administering Web Services*.
2. Use the `listWSMPolicySets` command to display a list of the policy sets in the repository.

```
listWSMPolicySets ([type=None])
```

You can limit the display to include only those policy sets that apply to a specific type of policy subject. To specify the type of subject, you must use the abbreviations specified in "Understanding Policy Subjects" in *Understanding Oracle Web Services Manager*.

For example, to display a list of policy sets that apply to SOAP web service endpoints:

```
wls:/jrfserver_domain/serverConfig>listWSMPolicySets('ws-service')
Global Policy Sets in Repository:

default-domain-ws-domain
GPAset1
```

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

4.4.4.2 Displaying the Configuration of a Policy Set

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" in *Administering Web Services*.
2. Use the `displayWSMPolicySet` command to display the configuration of a specified policy set.

```
displayWSMPolicySet ([name=None])
```

When you execute this command outside of a 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 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>displayWSPolicySet('default-domain-ws-domain')
```

Policy Set Details:

```
Display Name : default-domain-ws-domain
Type of Resources: SOAP Web Service
Scope of Resources: DOMAIN('*')
Description: Global policy attachments for Web Service Endpoint resources.
Enabled: true
Policy Reference: URI=oracle/wss_saml_or_username_token_service_policy,
category=security, enabled=true
reference.priority=10
```

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

4.4.4.3 Managing Sessions Using WLST

When using WLST to create, modify, and delete policy sets, you must execute the commands in the context of a session. Each session applies to a single policy subject, such as a policy set or a Fusion Middleware web service endpoint.

To create a session, use the `beginWSSession` command. After you have entered the desired commands, write the contents of the session to the repository using the `commitWSSession` command.

Use the `describeWSSession` command to describe the contents of the current session.

To exit a session without writing the contents, use the `abortWSSession` command.

Examples of these commands are provided in the subsequent sections. For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

4.4.4.4 Creating a New Policy Set 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" in *Administering Web Services*.
2. Begin a session using the `beginWSSession` command.

The `beginWSSession` 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 policy subject.

For example:

```
wls:/jrfserver_domain/serverConfig> beginWSSession()
```

Session started for modification.

3. Use the `createWSPolicySet` command to create a new, empty policy set. The `name`, `type`, and `attachTo` arguments are required.

```
createWSPolicySet(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 "[About Defining the Type and Scope of Resources for Globally Attached Policies](#)".

 **Note:**

When creating policy sets, the SOAP Web Service (`ws-service`) and SOAP Web Service Client (`ws-client`) subject types refer both to Oracle Infrastructure web services and clients and to Java EE web services and clients.

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 "[About Defining the Type and Scope of Resources for Globally Attached Policies](#)".

- `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> createWSMPolicySet ('all-domains-default-web-service-policies', 'ws-service', 'Domain("*")')
```

Description defaulted to "Global policy attachments for SOAP Web Service 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.

4. Specify a description using the `setWSMPolicySetDescription` command.

```
setWSMPolicySetDescription(description)
```

For example, to set the description as "Default policies for web services in any domain", use the following command:

```
wls:/jrfserver_domain/serverConfig> setWSMPolicySetDescription('Default policies for web services in any domain')
```

Description updated.

5. To attach a policy to the current policy set, first select the policy set, using `selectWSMPolicySet`, then use the `attachWSMPolicy` 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.

```
selectWSMPolicySet(policy set)
attachWSMPolicy(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 commands:

```
wls:/jrfserver_domain/serverConfig>selectWSMPolicySet('all-domains-default-web-service-policies')
```

The policy set is ready for modification in the session.

```
wls:/jrfserver_domain/serverConfig>attachWSMPolicy('oracle/wss11_saml_or_username_token_with_message_protection_service_policy')
```

```
Policy reference "oracle/wss11_saml_or_username_token_with_message_protection_service_policy" added.
```

6. Optionally, specify a configuration override or a run-time constraint. For details, refer to the following topics:
 - ["Overriding Configuration Properties for Globally Attached Policies Using WLST"](#).
 - ["Specifying Runtime Constraints in Policy Sets Using WLST"](#)
7. Optionally, display the configuration of the policy set during the current session using the `displayWSMPolicySet` command.

```
displayWSMPolicySet (name=None)
```

Note that when you execute this command within a 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>displayWSMPolicySet()
```

```
Policy Set Details:
```

```
-----
```

```
Display Name:      all-domains-default-web-service-policies
Type of Resources: SOAP Web Service
Scope of Resources: Domain("")
Description:       Default policies for web services in any domain
Enabled:          true
Policy Reference:  URI=oracle/
wss11_saml_or_username_token_with_message_protection_service_policy,
category=security, enabled=true, index=1
```

8. Validate the policy set using the `validateWSMPolicySet` command.

```
validateWSMPolicySet (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 session. If you do so, the `name` argument is required.

For example:

```
wls:/jrfserver_domain/serverConfig> validateWSMPolicySet()
```

The global policy set `all-domains-default-web-service-policies` is valid.

9. Write the contents of the current session to the repository using the `commitWSMSession` command.


```
wls:/jrfserver_domain/serverConfig> commitWSMSession()
```

The policy set all-domains-default-web-service-policies is valid.
Creating policy set all-domains-default-web-service-policies in repository.

Session committed successfully.

Alternately, you can choose to cancel any changes by using the `abortWSMSession` 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 *WLST Command Reference for Infrastructure Components*.

4.4.4.5 Cloning a Policy Set 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" in *Administering Web Services*.
2. Begin a session using the `beginWSMSession` command.

For example:

```
wls:/jrfserver_domain/serverConfig> beginWSMSession()
```

Session started for modification.

3. Use the `cloneWSMPolicySet` command to create a policy set using an existing policy set.

```
cloneWSMPolicySet(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 ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

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 `setWSMPolicySetScope` command.

- `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>cloneWSMPolicySet ('app-only-web-service-policies','all-domains-default-web-service-policies', None, 'Default policies for application jaxwsejb30ws')
```

The policy set was cloned successfully in the session.

Note that the `attachTo` argument was not specified in this example.

- Optionally, view the configuration of the policy set using the `displayWSMPolicySet` command.

For example:

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

Policy Set Details:
-----
Display Name:      app-only-web-service-policies
Type of Resources: SOAP Web Service
Scope of Resources: Domain("**")
Description:       Default policies for application jaxws-sut
Enabled:           true
Policy Reference:  URI=oracle/
wss11_saml_or_username_token_with_message_protection_service_policy,
category=security, enabled=true, index=1
```

- To change the resource scope of the attachments, use the `setWSMPolicySetScope` command.

```
setWSMPolicySetScope(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 more information, see ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

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> setWSMPolicySetScope
('Application("jaxwsejb30ws")')
```

Scope of resources updated.

- Optionally, specify a configuration override or a run-time constraint. For details, refer to the following topics:
 - ["Overriding Configuration Properties for Globally Attached Policies Using WLST"](#).
 - ["Specifying Runtime Constraints in Policy Sets Using WLST"](#)
- Optionally, view the configuration of the cloned policy set using the `displayWSMPolicySet` command.

For example:

```
wls:/jrfserver_domain/serverConfig>displayWSMPolicySet()

Policy Set Details:
-----
Display Name:      app-only-web-service-policies
Type of Resources: SOAP Web Service
Scope of Resources: Application("jaxwsejb30ws")
Description:       Default policies for application jaxwsejb30ws
Enabled:           true
Policy Reference:  URI=oracle/
wss11_saml_or_username_token_with_message_protection_service_policy,
category=security, enabled=true, index=1
```

- Write the contents of the current session to the repository using the `commitWSSMSession` command.

For example:

```
wls:/jrfserver_domain/serverConfig>commitWSMSession()
The policy set app-only-web-service-policies is valid.
Creating policy set app-only-web-service-policies in repository.

Session committed successfully.
```

Alternately, you can choose to cancel any changes by using the `abortWSMSession` 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 *WLST Command Reference for Infrastructure Components*.

4.4.4.6 Editing a Policy Set

You can edit a policy set using WLST:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Begin a session using the `beginWSMSession` command.

For example:

```
wls:/jrfserver_domain/serverConfig> beginWSMSession()

Session started for modification.
```

3. Use the `selectWSMPolicySet` command to select an existing policy set to edit.

```
selectWSMPolicySet(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> selectWSMPolicySet ('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 `attachWSMPolicy` command, identifying the policy by a specified URI using the `uri` argument.

```
attachWSMPolicy(uri)
```

For example, 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> attachWSMPolicy('oracle/
wss_saml_or_username_token_service_policy')
```

Policy reference oracle/wss_saml_or_username_token_service_policy added.

```
wls:/jrfServer_domain/serverConfig>attachWSMPolicy('oracle/log_policy')
```

Policy reference "oracle/log_policy" added.

- To remove policies from the policy set, use the `detachWSMPolicy` command, identifying the policy by a specified URI using the `uri` argument.

For example, 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> detachWSMPolicy('oracle/
wss11_saml_or_username_token_with_message_protection_service_policy')
```

```
Policy reference "oracle/
wss11_saml_or_username_token_with_message_protection_service_policy" removed.
```

- To enable or disable a policy attachment in the policy set, use the `enableWSMPolicy` command, identifying the policy by a specified URI using the `uri` argument.

```
enableWSMPolicy(uri, [enable=true])
```

The default is `true`.

To disable the `oracle/log_policy`, enter the following:

```
wls:/jrfServer_domain/serverConfig> enableWSMPolicy('oracle/log_policy', false)
```

```
Policy reference "oracle/log_policy" disabled.
```

5. Optionally, specify a configuration override or a run-time constraint. For details, refer to the following topics:
 - ["Overriding Configuration Properties for Globally Attached Policies Using WLST"](#).
 - ["Specifying Runtime Constraints in Policy Sets Using WLST"](#)
6. Validate the policy set using the `validateWSMPolicySet` command.

For example:

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

```
The global policy set all-domains-default-web-service-policies is valid.
```

7. Optionally, display the modified policy set using the `displayWSMPolicySet` command.

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

```
Policy Set Details:
```

```
-----
```

```
Display Name:      11-domains-default-web-service-policies
Type of Resources: SOAP Web Service
Scope of Resources: Domain("*")
Description:       Default policies for web services in any domain
Enabled:           true
Policy Reference:  URI=oracle/wss_saml_or_username_token_service_policy,
                  category=security, enabled=true, index=1
                  URI=oracle/log_policy, category=management, enabled=false,
index=2
```

8. To write the contents of the current session to the repository, use the `commitWSMSession` command.

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

```
The policy set all-domains-default-web-service-policies is valid.
```

```
Updating policy set all-domains-default-web-service-policies in repository.
Session committed successfully.
```

Alternately, you can choose to cancel any changes by using the `abortWSMSession` 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 *WLST Command Reference for Infrastructure Components*.

4.4.4.7 Validating a Policy Set

In addition to validating that the policy set adheres to the rules described in "[Validating Policy Attachments](#)", 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](#)".

For troubleshooting information, see "[Overview of Policy Attachment Issues Using WLST](#)".

4.4.4.8 Enabling and Disabling a Policy Set

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" in *Administering Web Services*.
2. Begin a session using the `beginWSMSession` command.

For example:

```
wls:/jrfserver_domain/serverConfig> beginWSMSession()
```

```
Session started for modification.
```

3. Specify the policy set to be modified using the `selectWSMPolicySet` command.

For example:

```
wls:/jrfServer_domain/serverConfig> selectWSMPolicySet ('all-domains-default-web-service-policies')
```

```
The policy set is ready for modification in the session.
```

4. Use the `enableWSMPolicySet` command to enable or disable a policy set.

```
enableWSMPolicySet([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> enableWSMPolicySet(false)

Policy set disabled.
```

5. Validate the policy set using the `validateWSMPolicySet` command.

For example:

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

The global policy set all-domains-default-web-service-policies is valid.
```

6. To write the contents of the current session to the repository, use the `commitWSMSession` command.

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

The policy set all-domains-default-web-service-policies is valid.
Updating policy set all-domains-default-web-service-policies in repository.

Session committed successfully.
```

Alternately, you can choose to cancel any changes by using the `abortWSMSession` 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 *WLST Command Reference for Infrastructure Components*.

4.4.4.9 Deleting Policy Sets Using WLST

You can use the following commands to delete policy sets in the repository:

- `deleteWSMPolicySet`—Deletes an individual policy set within the context of a session.
- `deleteWSMAllPolicySets`—Delete select or all policy sets in the repository. This command can be used inside or outside a session.

To delete an individual policy set in a session:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Begin a session using the `beginWSMSession` command.

For example:

```
wls:/jrfserver_domain/serverConfig> beginWSMSession()

Session started for modification.
```

3. Optionally, list the policy sets in the repository using the `listWSMPolicySets` command.

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

Global Policy Sets in Repository:
  app-only-web-service-policies
  all-domains-default-web-service-policies
```

4. Delete the desired policy set using the `deleteWSMPolicySet` command.

```
deleteWSMPolicySet (name)
```

For example:

```
wls:/jrfServer_domain/serverConfig> deleteWSMPolicySet('app-only-web-service-
policies')
```

The policy set was deleted successfully in the session.

5. Optionally, list the policy sets in the repository using the `listWSMPolicySets` command. Note that the policy set is flagged as delete pending.

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

```
Global Policy Sets in Repository:
  app-only-web-service-policies [delete pending]
  all-domains-default-web-service-policies
```

6. To write the contents of the current session to the repository, use the `commitWSMSession` command.

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

Deleting policy set app-only-web-service-policies from repository.

Session committed successfully.

Alternately, you can choose to cancel any changes by using the `abortWSMSession` 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" in *Administering Web Services*.
2. Optionally, list the policy sets in the repository using the `listWSMPolicySets` command.

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

```
Global Policy Sets in Repository:
  all-domains-default-web-service-policies
  ws-1
  ws-2
```

3. Delete the desired policy sets using the `deleteWSMAllPolicySets()` 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.

```
deleteWSMAllPolicySets(mode)
```

For example, to specify the policy sets to be deleted:

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

```
Starting Operation deleteWSMAllPolicySets ...
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.

```
deleteWSMAllPolicySets Operation Completed.
```

To force the deletion of all policy sets:

```
wls:/jrfServer_domain/serverConfig> deleteWSMAllPolicySets('force')

Starting Operation deleteWSMAllPolicySets ...

All policy sets were deleted successfully from repository.

deleteWSMAllPolicySets Operation Completed.
```

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

4.4.4.10 Specifying Runtime Constraints in Policy Sets Using WLST

You can specify a constraint in a policy set using the `setWSMPolicySetConstraint` command. This command can be used only during the creation or modification of a policy set within the context of a session.

The following procedure describes how to specify a run-time constraint while creating a new policy set, but you can also use the `setWSMPolicySetConstraint` command in a session while editing an existing policy set or creating a new 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" in *Administering Web Services*.
2. Begin a session using the `beginWSMSession` command.

For example:

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

Session started for modification.
```

3. Use the `createWSMPolicySet` 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>createWSMPolicySet('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 "[Creating a New Policy Set Using WLST](#)".

4. Attach a policy to the current policy set. First select the policy set, using `selectWSMPolicySet`, then use the `attachWSMPolicy` command, identifying the policy by a specified URI using the `uri` argument.

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>selectWSMPolicySet('domainExternal')
wls:/jrfServer_domain/serverConfig>attachWSMPolicy('oracle/
wss11_saml_or_username_token_with_message_protection_service_policy')
```

Policy reference "oracle/
wss11_saml_or_username_token_with_message_protection_service_policy" added.

5. Specify a run-time constraint using the `setWSMPolicySetConstraint(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>setWSMPolicySetConstraint('HTTPHeader("VIRTUAL_HOST_TYPE","External")')
```

Constraint updated.

6. Optionally, display the configuration of the policy set during the current session using the `displayWSMPolicySet` command.

For example:

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

Policy Set Details:

```
Display Name:      domainExternal
Type of Resources: SOAP Web Service
Scope of Resources: Domain("")
Constraint:        HTTPHeader("VIRTUAL_HOST_TYPE","External")
Description:       Authentication and message protection at domain scope for
external clients
Enabled:           true
Policy Reference:  URI=oracle/
wss11_saml_or_username_token_with_message_protection_service_policy,
category=security, enabled=true, index=1
```

7. Write the contents of the current session to the repository using the `commitWSMSession` command.

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

The policy set domainExternal is valid.
Creating policy set domainExternal in repository.
Session committed successfully.

4.4.5 Viewing Policies Attached to a Web Service with WLST

You can use WLST commands to view the policies that are attached to a web service.

To do so, complete the following steps:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Use the `listWSMPolicySubjects` WLST command to display a list of the web services in your application as described in "Viewing the Web Services in an Application Deployment Using Fusion Middleware Control" *Administering Web Services*
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 `Wsd1ConcreteService`:

```
wls:/wls-domain/serverConfig> listWebServicePorts ("/base-domain/AdminServer/
jaxwsejb30ws", "jaxwsejb","web","WsdConcreteService")
```

```
WsdConcretePort http://host.example.com:7001/jaxwsejb/WsdAbstract
```

4. Use the `listWebServicePolicies` command to view the policies that are attached to a web service port.

```
listWebServicePolicies(application,moduleOrCompName,moduleType,serviceName,subjectName)
```

For example, to view the policies attached to the `WsdConcretePort` port and any policy override settings:

```
wls:/wls_domain/serverConfig> listWebServicePolicies("/jaxwsejb30ws",
"jaxwsejb","web","WsdConcreteService","WsdConcretePort")
```

```
WsdConcretePort :
    URI="oracle/mex_request_processing_service_policy",
category=wsconfig, policy-status=enabled; source=local policy set; reference-
status=enabled; effective=true
    URI="oracle/mtom_encode_fault_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
    URI="oracle/max_request_size_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="max.request.size", value="-1"
    URI="oracle/request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
    URI="oracle/soap_request_processing_service_policy",
category=wsconfig, policy-status=enabled; source=local policy set; reference-
status=enabled; effective=true
    URI="oracle/ws_logging_level_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="logging.level", value=""
    URI="oracle/test_page_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
    URI="oracle/wsd_request_processing_service_policy",
category=wsconfig, policy-status=enabled; source=local policy set; reference-
status=enabled; effective=true
    URI="oracle/
wss11_saml_or_username_token_with_message_protection_service_policy",
category=security, policy-status=enabled; source=global policy set "domainExternal",
scope="Domain(*)";reference-status=enabled; effective=true
```

The web service is secure in this context.

4.4.6 Displaying the Effective Policy Set Using WLST

The `displayWSMEffectivePolicySet()` command allows you to display the configuration of the effective policy set corresponding to the policy subject. It displays the configuration of the actual runtime policy set used at the time of policy enforcement.

The `displayWSMEffectivePolicySet()` command also displays the global policy attachment information. This policy set and global policy attachment information is stored within the policy subject.

Compare this command with the `displayWSMPolicySet` command, which displays only the selected global policy set or the selected direct policy set, or with the

`previewWSMEffectivePolicySet`, which displays the effective policy set, including changes made to the actual runtime policy set, within the current session.

The changes that you make in the course of the session will not show up in the response to the `displayWSMEffectivePolicySet()` command until you commit the session.

In the following example, the `WsdConcreteService#WsdConcretePort` endpoint belonging to the `jaxwsejb30ws` application is selected. The `displayWSMEffectivePolicySet()` command displays the security policies that are in effect for the endpoint and their status.

```
wls:/jrfServer_domain/serverConfig> selectWSMPolicySubject('jaxwsejb30ws',
'#jaxwsejb', 'WS-Service({http://www.oracle.com/jaxws/tests/
concrete}WsdConcreteService#WsdConcretePort)')
```

The policy subject is selected for modification.

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

```
Context : Constraint="HTTPHeader('VIRTUAL_HOST_TYPE','External')"
        URI="oracle/mex_request_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
        URI="oracle/mtom_encode_fault_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
        URI="oracle/max_request_size_policy", category=wsconfig, policy-status=enabled;
source=local policy set; reference-status=enabled; effective=true
        Property name="max.request.size", value="-1"
        URI="oracle/request_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
        URI="oracle/soap_request_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
        URI="oracle/ws_logging_level_policy", category=wsconfig, policy-status=enabled;
source=local policy set; reference-status=enabled; effective=true
        Property name="logging.level", value=""
        URI="oracle/test_page_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
        URI="oracle/wsd_request_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
        URI="oracle/
wss11_saml_or_username_token_with_message_protection_service_policy", category=security,
policy-status=enabled; source=global policy set "domainExternal", scope="Domain(*)";
reference-status=enabled; effective=true
```

The web service is secure in this context.

The `oracle/mex_request_processing_service_policy` and `oracle/mtom_encode_fault_service_policy` policies are detached from the service:

```
wls:/base_domain/serverConfig> detachWSMPolicies(['oracle/
mex_request_processing_service_policy', 'oracle/mtom_encode_fault_service_policy'])
```

```
Policy reference "oracle/mex_request_processing_service_policy" removed.
Policy reference "oracle/mtom_encode_fault_service_policy" removed.
```

If you now run the `previewWSMEffectivePolicySet()` command, you will see that the two policies have been removed from the policy set for the `WsdConcreteService#WsdConcretePort` endpoint.

```
wls:/base_domain/serverConfig> previewWSMEffectivePolicySet()
```

```
Context : Constraint="HTTPHeader('VIRTUAL_HOST_TYPE','External')"
        URI="oracle/
wss11_saml_or_username_token_with_message_protection_service_policy", category=security,
```

```

policy-status=enabled; source=global policy set "domainExternal", scope="Domain(*)";
reference-status=enabled; effective=true
    URI="oracle/max_request_size_policy", category=wsconfig, policy-status=enabled;
source=local policy set; reference-status=enabled; effective=true
    Property name="max.request.size", value="-1"
    URI="oracle/request_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
    URI="oracle/soap_request_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
    URI="oracle/ws_logging_level_policy", category=wsconfig, policy-status=enabled;
source=local policy set; reference-status=enabled; effective=true
    Property name="logging.level", value=""
    URI="oracle/test_page_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
    URI="oracle/wsd1_request_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true

```

The web service is secure in this context.

If you run the `displayWSMEffectivePolicySet()` command again, the `oracle/mex_request_processing_service_policy` and `oracle/mtom_encode_fault_service_policy` policies will appear in the response. The policies will not be removed from the response until you issue the `commitWSMSession()` command.

4.5 About 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.

The predefined policies required to secure servlet applications, such as ADF business components exposed as RESTful servlets are defined in [OWSM Policies Supported for RESTful Web Services and Clients](#). For more information about these policies and how to manually configure them, see [Oracle Web Services Manager Predefined Policies](#).

For servlet applications, the OWSM 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"](#)
- ["Attaching Policies Globally to Servlet Applications"](#)

4.5.1 Attaching Policies Directly to Servlet Applications

To attach policies directly to servlet applications, you must modify the `web.xml` deployment descriptor file to define the OWSM servlet filter, associate it with a servlet to be secured, and define the policy attachment metadata.

You can map an OWSM 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 OWSM security filter by adding a `<filter>` element, and defining the following subelements:

- a. Specify a meaningful name for the OWSM servlet filter using the `<filter-name>` element.

For example:

```
<filter>
<filter-name>OWSM Security Filter</filter-name>
```

- b. Define the OWSM 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 OWSM 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 [Schema Reference for Web Services 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 OWSM 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>/testervlet</url-pattern>
</servlet-mapping>
```

4. Repeat steps 1 through 3 for each servlet you wish to secure.

The following example shows how to update the `web.xml` file to attach policies to a servlet application.

```
<?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[<scall:policySet name="policySet"
        appliesTo="REST-Resource()"
        attachTo="Service('*')"
        xmlns:scall="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>
        </scall: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>/testervlet</url-pattern>
    </servlet-mapping>
</web-app>

```

4.5.2 Attaching Policies Globally to Servlet Applications

To attach policies globally to servlet applications, you create and attach a policy set using WLST.

For more information, see ["About Attaching Policies Globally Using WLST"](#).

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.

To attach policies globally to servlet applications:

1. Define the OWSM security filter by adding a `<filter>` element, and defining the following subelements:

- a. Specify a meaningful name for the OWSM servlet filter using the `<filter-name>` element.

For example:

```

<filter>
  <filter-name>OWSM Security Filter</filter-name>

```

- b. Define the OWSM 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 OWSM 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 OWSM 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>/testervlet</url-pattern>
</servlet-mapping>
```

4. Repeat steps 1 through 3 for each servlet you wish to secure.

The following example shows 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> beginWSSession()
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean as the root.
For more help, use help('domainRuntime')

Session started for modification.
wls:/my_domain/serverConfig> createWSPolicySet('all-domains-default-REST','REST-Resource',
'Domain(*)')

Description defaulted to "Global policy attachments for Java EE RESTful Resource resources."
The policy set was created successfully in the session.

wls:/my_domain/serverConfig>selectWSPolicySet('all-domains-default-REST')

The policy set is ready for modification in the session.

wls:/my_domain/serverConfig> attachWSPolicy('oracle/http_basic_auth_over_ssl_service_policy')

Policy reference "oracle/http_basic_auth_over_ssl_service_policy" added.

wls:/my_domain/serverConfig> commitWSSession()

The policy set all-domains-default-REST is valid.
Creating policy set all domains-default-REST in repository.

Session committed successfully.

wls:/my_domain/serverConfig> displayWSPolicySet('all-domains-default-REST')

Policy Set Details:
-----
Display Name : all-domains-default-REST
Type of Resources: RESTful Resource
Scope of Resources: Domain(*)
Description: Global policy attachments for Java EE RESTful Resource resources.
Enabled: true
Policy Reference: URI=oracle/wss_http_token_service_policy, category=security, enabled=true
```



```
wls:/my_domain/serverConfig>
```

4.6 About Securing the URI patterns for Resources in RESTful Web Services

For RESTful web services built using Jersey 2.x JAX-RS RI or Jersey 1.x JAX-RS RI, you can create OWSM policies globally to secure the resources or services that are part of an application.

A RESTful application includes multiple root resources, sub-resources, sub-resource locators, and sub-resource method. These resources use URLs to specify their locations. With this feature, you can secure the application sub components by configuring the URL Patterns.

For example, a RESTful application may require both user authentication and anonymous access of certain data. OWSM provides you the flexibility to secure an entire application or part of the application. You can do this by securing the URI patterns of the module, service, resource path, resource Sub-path, or HTTP method in an application.

OWSM also supports regular expression for the URL pattern (`PATH` and `METHOD`) while defining GPA policy sets.



Note:

You can use local policy attachment to secure the resource path. OWSM does not support local policy attachment to secure the resource Sub-path. You cannot use the local policy attachment to secure the URI patterns using a path or HTTP method.

See: [Example Scenarios: Creating Policies to Secure URI patterns Using WLST](#)

4.6.1 Example Scenario: Creating Policies to Secure URI patterns

These example scenarios shows how to create policies using the WLST command for the RESTful Web Services and its module, service, resource path, resource sub-path, or HTTP method in an application.

You can also use the Fusion Middleware Control to create policies. See [Creating a Policy Set Using Fusion Middleware Control](#).

Topics:

- [Creating a Policy Set to Secure the Application](#)
- [Creating a Policy Set to Secure the Module in an Application](#)
- [Creating Policy Sets to Secure Multiple Modules with Different Policies in the Application](#)
- [Creating a Policy Set to Secure the Paths for a Module in the Service](#)
- [Creating Policy Sets to Grant Anonymous Access for a Specific Path in a Secured Application](#)
- [Creating a Policy Set to Secure the HTTP Method in the Application](#)

4.6.1.1 Creating a Policy Set to Secure the Application

This scenario illustrates how to creating a new policy set using the WLST command to secure the entire application for a JAX-WS web service. The JAX-WS web service uses the `oracle/multi_token_rest_service_policy` policy for authentication.

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Begin a session using the `beginWSMSession` command.

The `beginWSMSession` 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 policy subject.

For example:

```
wls:/jrfserver_domain/serverConfig> beginWSMSession()
```

```
Session started for modification.
```

3. Use the `createWSMPolicySet` command to create a new policy set:

```
createWSMPolicySet(name,type,attachTo,[description=None],[enable='true'])
```

Where:

- `name`: Specify the name of the new, empty policy set. For example, `all-domains-default-web-service-policies`.
- `type`: Specify the type of policy subject to which the new policy set applies. For example, `rest-resource`.
- `attachTo`: Specify the application (`Application("expression")`) to which the policy set will be attached. For example, `APPLICATION("jaxrsservices")`. See ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

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 ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

- `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:

```
wls:/jrfserver_domain/serverConfig> createWSMPolicySet ('all-domains-default-web-service-policies', 'rest-resource', 'APPLICATION("jaxrsservices")')
```

4. Use the `attachWSMPolicy` command to attach the `oracle/multi_token_rest_service_policy` policy.

```
wls:/jrfserver_domain/serverConfig>selectWSMPolicySet('all-domains-default-web-service-policies')
```

The policy set is ready for modification in the session.

```
wls:/jrfserver_domain/serverConfig>attachWSMPolicy('oracle/multi_token_rest_service_policy')
```

```
Policy reference "oracle/multi_token_rest_service_policy" added.
```

4.6.1.2 Creating a Policy Set to Secure the Module in an Application

This scenario illustrates how to creating a new policy set to secure a module (WAR file) in the application. The module uses the `oracle/multi_token_rest_service_policy` policy for authentication.

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Begin a session using the `beginWSMSession` command.

The `beginWSMSession` 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 policy subject.

For example:

```
wls:/jrfserver_domain/serverConfig> beginWSMSession()
```

```
Session started for modification.
```

3. Use the `createWSMPolicySet` command to create a new policy set.

```
createWSMPolicySet(name,type,attachTo,[description=None],[enable='true'])
```

Where:

- `name`: Specify the name of the new, empty policy set. For example, `all-domains-default-web-service-policies`.
- `type`: Specify the type of policy subject to which the new policy set applies. For example, `rest-resource`.
- `attachTo`: Specify the module (`Module("expression")`) to which the policy set will be attached. For example, `MODULE("Module1")`. See ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

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 ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

- `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 the module in an application:

```
wls:/jrfserver_domain/serverConfig> createWSMPolicySet ('all-domains-default-web-  
service-policies', 'rest-resource', 'APPLICATION("jaxrsservices") and  
MODULE("Module1")')
```

4. To attach a policy to the current policy set, first select the policy set, using `selectWSMPolicySet`, then use the `attachWSMPolicy` 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.

```
selectWSMPolicySet(policy set)  
attachWSMPolicy(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 commands:

```
wls:/jrfserver_domain/serverConfig>selectWSPolicySet('all-domains-default-web-service-policies')
```

The policy set is ready for modification in the session.

```
wls:/jrfserver_domain/serverConfig>attachWSPolicy('oracle/multi_token_rest_service_policy')
```

Policy reference "oracle/multi_token_rest_service_policy" added.

5. Optionally, specify a configuration override or a run-time constraint. For details, refer to the following topics:
 - ["Overriding Configuration Properties for Globally Attached Policies Using WLST"](#).
 - ["Specifying Runtime Constraints in Policy Sets Using WLST"](#)

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

4.6.1.3 Creating Policy Sets to Secure Multiple Modules with Different Policies in the Application

This scenario illustrates how to create new policy sets using WLST command to secure two modules (Module1 and Module2) for the application in the domain. The modules uses different policies for authentication. In this example, Module1 is using the oracle/multi_token_rest_service_policy policy for authentication and Module2 is using the oracle/http_jwt_token_service_policy policy for authentication.

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Begin a session using the `beginWSMSession` command.

The `beginWSMSession` 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 policy subject.

For example:

```
wls:/jrfserver_domain/serverConfig> beginWSMSession()
```

Session started for modification.

3. Use the `createWSPolicySet` command to create a new policy set for Module1.

```
createWSPolicySet (name,type,attachTo,[description=None],[enable='true'])
```

Where:

- `name`: Specify the name of the new, empty policy set. For example, `all-domains-default-web-service-policies`.
- `type`: Specify the type of policy subject to which the new policy set applies. For example, `rest-resource`.

- `attachTo`: Specify the application (`Module("expression")`) to which the policy set will be attached. For example, `MODULE("Module1")`. See ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

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 ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

- `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 the `Module1` in an application:

```
wls:/jrfserver_domain/serverConfig> createWSMPolicySet ('all-domains-default-web-  
service-policies', 'rest-resource', 'APPLICATION("jaxrservices") and  
MODULE("Module1")')
```

4. Follow the above step to create a new policy set for `Module2`.

```
wls:/jrfserver_domain/serverConfig> createWSMPolicySet ('all-domains-default-web-  
service-policies', 'rest-resource', 'APPLICATION("jaxrservices") and  
MODULE("Module2")')
```

5. Use the `attachWSMPolicy` command to attach the `oracle/
multi_token_rest_service_policy` policy for `Module1`.

```
wls:/jrfserver_domain/serverConfig>selectWSMPolicySet('all-domains-default-web-  
service-policies')
```

The policy set is ready for modification in the session.

```
wls:/jrfserver_domain/serverConfig>attachWSMPolicy('oracle/  
multi_token_rest_service_policy')
```

Policy reference "oracle/multi_token_rest_service_policy" added.

6. Use the `attachWSMPolicy` command to attach the `oracle/
http_jwt_token_service_policy` policy for `Module2`.

```
wls:/jrfserver_domain/serverConfig>selectWSMPolicySet('all-domains-default-web-  
service-policies')
```

The policy set is ready for modification in the session.

```
wls:/jrfserver_domain/serverConfig>attachWSMPolicy('oracle/  
http_jwt_token_service_policy')
```

Policy reference "oracle/http_jwt_token_service_policy" added.

4.6.1.4 Creating a Policy Set to Secure the Paths for a Module in the Service

This scenario illustrates how to create a new policy set using the WLST command to secure all the paths for a module in the service. The paths use `oracle/
multi_token_rest_service_policy` policy for authentication.

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Begin a session using the `beginWSMSession` command.

The `beginWSMSession` 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 policy subject.

For example:

```
wls:/jrfserver_domain/serverConfig> beginWSMSession()
```

Session started for modification.

3. Use the `createWSMPolicySet` command to create a new policy set:

```
createWSMPolicySet(name,type,attachTo,[description=None],[enable='true'])
```

Where:

- `name` :Specify the name of the new, empty policy set. For example, `all-domains-default-web-service-policies`.
- `type`: Specify the type of policy subject to which the new policy set applies. For example, `rest-resource`.
- `attachTo` :Specify the paths for the module in the service (`Path("expression")`) to which the policy set will be attached. For example, `PATH("Module1/Module1Service1App/*.*)` . See ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

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 ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

- `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:

```
wls:/jrfserver_domain/serverConfig> createWSMPolicySet ('all-domains-default-web-service-policies', 'rest-resource', 'Application("jaxrservices") and PATH("Module1/Module1Service1App/*.*)')
```

4. Use the `attachWSMPolicy` command to attach the `oracle/multi_token_rest_service_policy` policy.

```
wls:/jrfserver_domain/serverConfig>selectWSMPolicySet('all-domains-default-web-service-policies')
```

The policy set is ready for modification in the session.

```
wls:/jrfserver_domain/serverConfig>attachWSMPolicy('oracle/multi_token_rest_service_policy')
```

Policy reference "oracle/multi_token_rest_service_policy" added.

4.6.1.5 Creating Policy Sets to Grant Anonymous Access for a Specific Path in a Secured Application

This scenario illustrates how to creating a new policy set using the WLST command to secure the entire application for a JAX-WS web service and provide anonymous access for a specific path. The JAX-WS web service uses the `oracle/multi_token_rest_service_policy` policy for authentication and the path uses the `oracle/no_authentication_service_policy` for

anonymous access. In this scenario we secure the "Hello World!" application and provide anonymous access for paths starting with `/helloworld`.

In this scenario, the "Hello World!" application is used as an example where the entire application is secured and anonymous access is granted for paths starting with `/helloworld`.

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Begin a session using the `beginWSMSession` command.

The `beginWSMSession` 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 policy subject.

For example:

```
wls:/jrfserver_domain/serverConfig> beginWSMSession()
```

Session started for modification.

3. Use the `createWSMPolicySet` command to create a new policy set:

```
createWSMPolicySet(name,type,attachTo,[description=None],[enable='true'])
```

Where:

- `name`: Specify the name of the new, empty policy set. For example, `all-domains-default-web-service-policies`.
- `type`: Specify the type of policy subject to which the new policy set applies. For example, `rest-resource`.
- `attachTo`: Specify the application (`Application("expression")`) to which the policy set will be attached. For example, `APPLICATION("jaxrsservices")`. See ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

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 ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

- `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 the JAX-WS web service in a domain:

```
wls:/jrfserver_domain/serverConfig> createWSMPolicySet ('all-domains-default-web-service-policies', 'rest-resource', 'APPLICATION("jaxrsservices")')
```

4. Use the `createWSMPolicySet` command to create a policy set to grant anonymous access for paths starting with `/helloworld`.

```
createWSMPolicySet(name,type,attachTo,[description=None],[enable='true'])
```

Where:

- `name`: Specify the name of the new, empty policy set. For example, `all-domains-default-web-service-policies`.
- `type`: Specify the type of policy subject to which the new policy set applies. For example, `rest-resource`.

- `attachTo`: Specify the path (`Path("expression")`) to which the policy set will be attached. For example, `PATH(".*/*helloworld/*.*")`. See "About Defining the Type and Scope of Resources for Globally Attached Policies".

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 "About Defining the Type and Scope of Resources for Globally Attached Policies".

- `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:

```
wls:/jrfserver_domain/serverConfig> createWSMPolicySet ('all-domains-default-web-  
service-policies', 'rest-resource', 'APPLICATION("jaxrsservices") and PATH(".*/  
helloworld/*.*")')
```

5. Use the `attachWSMPolicy` command to attach the `oracle/multi_token_rest_service_policy` policy for the application.

```
wls:/jrfserver_domain/serverConfig>selectWSMPolicySet('all-domains-default-web-  
service-policies')
```

The policy set is ready for modification in the session.

```
wls:/jrfserver_domain/serverConfig>attachWSMPolicy('oracle/  
multi_token_rest_service_policy')
```

Policy reference "oracle/multi_token_rest_service_policy" added.

6. Use the `attachWSMPolicy` command to attach the `oracle/no_authentication_service_policy` policy for `PATH(".*/*helloworld/*.*")`.

```
wls:/jrfserver_domain/serverConfig>selectWSMPolicySet('all-domains-default-web-  
service-policies')
```

The policy set is ready for modification in the session.

```
wls:/jrfserver_domain/serverConfig>attachWSMPolicy('oracle/  
no_authentication_service_policy')
```

Policy reference "oracle/http_jwt_token_service_policy" added.

4.6.1.6 Creating a Policy Set to Secure the HTTP Method in the Application

This scenario illustrates how to create a new policy set using the WLST command to secure the `POST` REST method for a service. The HTTP method uses the `oracle/multi_token_rest_service_policy` policy for authentication.

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Begin a session using the `beginWSMSession` command.

The `beginWSMSession` 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 policy subject.

For example:


```
wls:/jrfserver_domain/serverConfig> beginWSMSession()
```

Session started for modification.

3. Use the `createWSMPolicySet` command to create a new, empty policy set. The `name`, `type`, and `attachTo` arguments are required.

```
createWSMPolicySet(name, type, attachTo, [description=None], [enable='true'])
```

Where:

- `name` : Specify the name of the new, empty policy set. For example, `all-domains-default-web-service-policies`.
- `type` : Specify the type of policy subject to which the new policy set applies. For example, `rest-resource`.
- `attachTo` : Specify the HTTP method (`Method("expression")`) to which the policy set will be attached. For example, `Method("POST")`.

This argument must use a supported expression that defines a valid resource scope in a supported format. See ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

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 ["About Defining the Type and Scope of Resources for Globally Attached Policies"](#).

- `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 the `POST` REST method for all services in a domain:

```
wls:/jrfserver_domain/serverConfig> createWSMPolicySet ('all-domains-default-web-service-policies', 'rest-resource', 'APPLICATION("jaxrsservices") and Method("POST")')
```

4. Use the `attachWSMPolicy` command to attach the `oracle/multi_token_rest_service_policy` policy.

```
wls:/jrfserver_domain/serverConfig>selectWSMPolicySet('all-domains-default-web-service-policies')
```

The policy set is ready for modification in the session.

```
wls:/jrfserver_domain/serverConfig>attachWSMPolicy('oracle/multi_token_rest_service_policy')
```

Policy reference "oracle/multi_token_rest_service_policy" added.

4.7 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:**

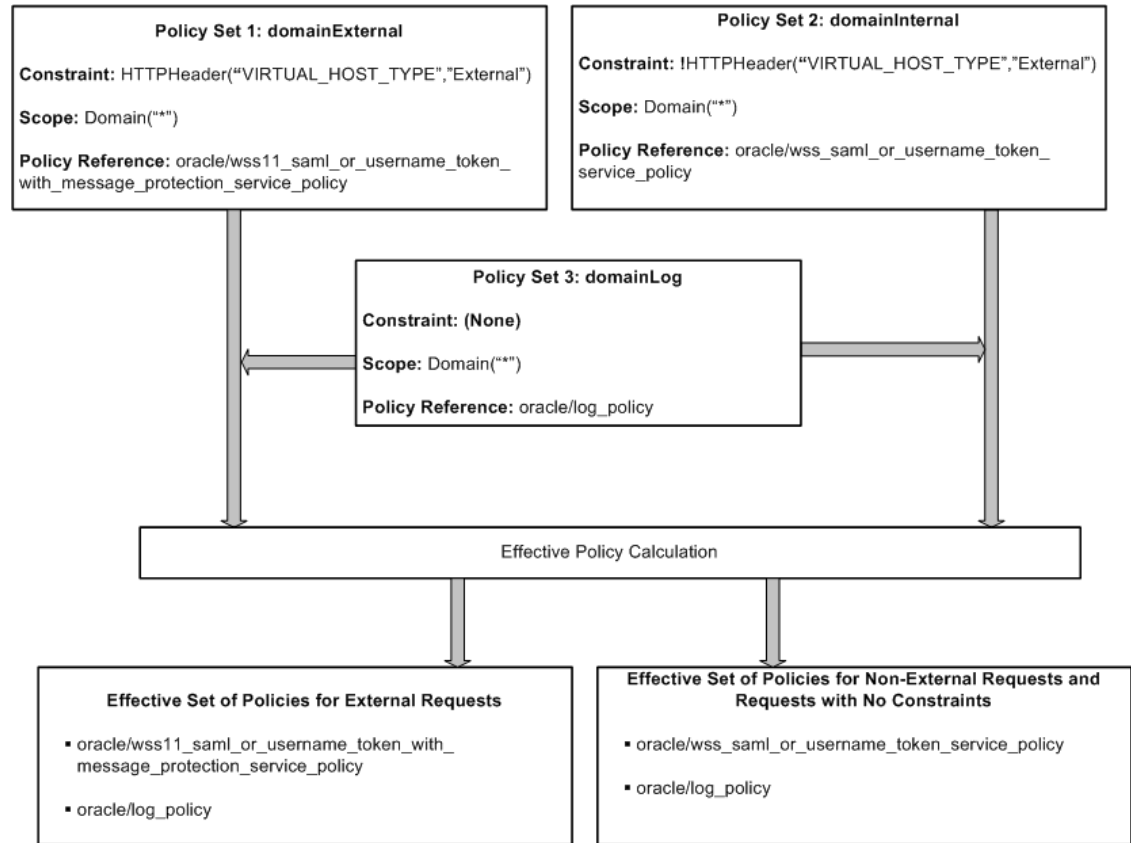
The run-time constraint function `HTTPHeader` is only certified to use when Oracle HTTP Server is configured at the front end *and* the Oracle HTTP Server administrator has added a custom `VIRTUAL_HOST_TYPE` header to the request. For details about adding the header to the request, see "[Configuring the Oracle HTTP Server to Specify the Request Origin](#)".

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](#)".
- 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 4-14](#) illustrates the effective policies for external and non-external requests determined using constraints in three different policy sets.

Figure 4-14 Effective Policy Calculation for Policy Sets with Run Time Constraints



For details about how to specify a run-time constraint in a policy set, refer to the following topics:

- ["Specifying Run-time Constraints in a Policy Set Using Fusion Middleware Control"](#).
- ["Specifying Runtime Constraints in Policy Sets Using WLST"](#).

4.8 About Defining the Type and Scope of Resources for Globally Attached Policies

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.

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 identifies the location within the resource where the policy set containing policies should be applied. It is mapped to the `attachTo` attribute of the policy set and is used for conflict resolution when multiple policy sets exist.

- [Defining the Resource Type](#)
- [Defining the Resource Scope](#)
- [Determining the Namespace for a Web Service](#)

- [Examples of Creating Policy Sets Using Different Resource Types and Scopes](#)

4.8.1 Defining the 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.

For a list of valid resource types and their WLST equivalents, see "Understanding Policy Subjects" in *Understanding Oracle Web Services Manager*.



Note:

When creating policy sets, the SOAP Web Service and SOAP Web Service Client subject types refer both to Oracle Infrastructure web services and clients and to Java EE web services and clients.

4.8.2 Defining the 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.

For a list resource scopes that are valid for each policy subject type, see "Understanding Policy Subjects" in *Understanding Oracle Web Services Manager*.

When specifying the resource scope in WLST, you need to use a supported expression for each scope. These expressions are required for the following arguments:

- `attachTo` argument of the `createWSMPolicySet` and `cloneWSMPolicySet` commands
- `expression` argument of the `setWSMPolicySetScope` 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.

[Table 4-2](#) describes the supported expressions for WLST, and the resource scope name as specified in Fusion Middleware Control.

Table 4-2 Supported Expressions for the Resource Scope

Fusion Middleware Control Resource Scope Name	Supported Expression in WLST	Description
Domain Name	Domain("expression")	This value will be matched against a policy subject based on the management domain in which it is deployed.
Application Name	Application("expression")	This value will be matched against a policy subject based on the name of the application in which it is located.
Partition Name	Partition("expression")	This value will be matched against a policy subject based on the name of the SOA partition in which it is located.
Application Module Name or Connection Name	Module("expression")	This value will be matched against a policy subject based on the name of the application module or connection in which it is located.
SOA Composite Name	Composite("expression")	<p>This value will be matched against a policy subject based on the name of the SOA composite in which it is located.</p> <p>Note: For a composite, the expression should use the composite name only, for example: <code>Composite("*Basic_SOA_Client*")</code></p> <p>Do not include the SOA partition or composite revision number in the expression.</p>
ESS Job Name	Jobname("expression")	This value will be matched against a policy subject based on the name of the Oracle Enterprise Scheduler web service job in which it is located.
Resource Path	Path("expression")	<p>This value will be matched against a policy subject based on the name of the resource path in which it is located.</p> <p>For RESTful web service, resource path is the request URI associated with the URL of the request sent by the client. The value of the resource path is the portion of the URL address that immediately follows the host name and port and port number, including the context root (module name) and excludes any query parameters.</p> <p>To specify multiple resource paths for a RESTful web service, use the pipe in path expression. For example:</p> <pre>PATH('sample/app/path2/devices.* sample/app/path1/.*')</pre>

Table 4-2 (Cont.) Supported Expressions for the Resource Scope

Fusion Middleware Control Resource Scope Name	Supported Expression in WLST	Description
HTTP Method Name	Method("expression")	This value will be matched against a policy subject based on the name of the HTTP method in the resource path in which it is located. For example: <code>PATH('sample/app/path/devices')</code> and <code>METHOD('POST')</code>
Reference or Web Service Client Name	Reference("expression")	This value will be matched against a policy subject based on the name of the reference or web service client in which it is located.
RESTful Application, Service, or Web Service Endpoint Name	Service("expression")	This value will be matched against a policy subject based on the name of the service, RESTful application, or web service endpoint in which it is located. Note: For a service, the expression must include the namespace and the service name, for example: <code>Service("{http://mynamespace/}myService")</code> For applications assembled prior to PS5, the namespace is not displayed in the <code>listWSMPolicySubjects</code> 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" .
SOA Component Name	Component("expression")	This value will be matched against a policy subject based on the name of the SOA component in which it is located.
Port Name	Port("expression")	This value will be matched against a policy subject based on the name of the port in which it is located.
Java EE Web Service Client EJB Name	EJBName("expression")	This value will be matched against a policy subject based on the name of the Java EE web service client EJB in which it is located.
Callback Interface Name	PortType("expression")	This value will be matched against a policy subject based on the name of the callback interface in which it is located.

4.8.3 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, 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 "Viewing the Web Service WSDL Document" in *Administering Web Services*.
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
```

4.8.4 Examples of Creating Policy Sets Using Different Resource Types and Scopes

The following examples demonstrate how to create policy sets using different resource types and scopes.

The following example 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`).

```
beginWSMSession()
createWSMPolicySet('Async callback client', 'ws-callback', 'Domain("FinancialDomain")
and Application("Expense*")',
'Global policy for asynchronous callback client', true)
selectWSMPolicySet('Async callback client')
attachWSMPolicy('oracle/wss10_saml_token_client_policy')
validateWSMPolicySet()
commitWSMSession()
displayWSMPolicySet('Async callback client')
```

The following example creates a policy set named `web_connection_cost_service` for an ADF SOAP 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`).

```
beginWSMSession()
createWSMPolicySet('web_connection_cost_service', 'ws-connection', 'Domain("SCMDomain")
and Application("ScmCst*") and Module("*Costs")', enable=true)
selectWSMPolicySet('web_connection_cost_service')
attachWSMPolicy('oracle/wss10_saml_token_client_policy')
validateWSMPolicySet()
commitWSMSession()
displayWSMPolicySet('web_connection_cost_service')
```

4.9 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

Note:

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" in *Administering Web Services*.
2. Migrate the attachments using the `migrateWSMAttachments` 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`.

```
migrateWSMAttachments(mode='prompt')
```

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> migrateWSMPolicyAttachments()
```

```
-----
Application:      /WLS/base_domain/jaxwsejb30ws
Assembly:        #jaxwsejb
Subject:         WS-Service({http://www.oracle.com/jaxws/tests/
concrete}WsdConcreteService#WsdConcretePort)
```



```

Policy Reference:  URI=oracle/wss_saml_or_username_token_service_policy,
source=global policy set "default-domain-ws-domain", reference-status=enabled
                    reference.priority=10
                    URI=oracle/mex_request_processing_service_policy, source=local
policy set, reference-status=enabled
                    URI=oracle/mtom_encode_fault_service_policy, source=local
policy set, reference-status=enabled
                    URI=oracle/max_request_size_policy, source=local policy set,
reference-status=enabled
                    max.request.size=-1
                    URI=oracle/request_processing_service_policy, source=local
policy set, reference-status=enabled
                    URI=oracle/soap_request_processing_service_policy, source=local
policy set, reference-status=enabled
                    URI=oracle/ws_logging_level_policy, source=local policy set,
reference-status=enabled
                    logging.level=
                    URI=oracle/test_page_processing_service_policy, source=local
policy set, reference-status=enabled
                    URI=oracle/wSDL_request_processing_service_policy, source=local
policy set, reference-status=enabled
                    URI=oracle/
wss10_saml20_token_with_message_protection_service_policy, source=local policy set,
reference-status=disabled
-----
Application:      /WLS/base_domain/jaxwsejb30ws
Assembly:        #jaxwsejb
Subject:         WS-Service({http://soapinterop.org/
DoclitWrapperWTJ}DoclitWrapperWTJService#DoclitWrapperWTJPort)
Policy Reference: URI=oracle/wss_saml_or_username_token_service_policy,
source=global policy set "default-domain-ws-domain", reference-status=enabled
                    reference.priority=10
                    URI=oracle/mex_request_processing_service_policy, source=local
policy set, reference-status=enabled
                    URI=oracle/mtom_encode_fault_service_policy, source=local
policy set, reference-status=enabled
                    URI=oracle/max_request_size_policy, source=local policy set,
reference-status=enabled
                    max.request.size=-1
                    URI=oracle/request_processing_service_policy, source=local
policy set, reference-status=enabled
                    URI=oracle/soap_request_processing_service_policy, source=local
policy set, reference-status=enabled
                    URI=oracle/ws_logging_level_policy, source=local policy set,
reference-status=enabled
                    logging.level=
                    URI=oracle/test_page_processing_service_policy, source=local
policy set, reference-status=enabled
                    URI=oracle/wSDL_request_processing_service_policy, source=local
policy set, reference-status=enabled
                    URI=oracle/
wss10_username_token_with_message_protection_service_policy, source=local policy
set, reference-status=enabled
-----
Application:      /WLS/base_domain/jaxwsejb30ws
Assembly:        #jaxwsejb
Subject:         WS-Service({http://www.oracle.com/jaxws/
tests}CalculatorService#CalculatorPort)
Policy Reference: URI=oracle/wss_saml_or_username_token_service_policy,
source=global policy set "default-domain-ws-domain", reference-status=enabled
                    reference.priority=10
                    URI=oracle/mex_request_processing_service_policy, source=local

```

```

policy set, reference-status=enabled
    URI=oracle/mtom_encode_fault_service_policy, source=local
policy set, reference-status=enabled
    URI=oracle/max_request_size_policy, source=local policy set,
reference-status=enabled
    max.request.size=-1
    URI=oracle/request_processing_service_policy, source=local
policy set, reference-status=enabled
    URI=oracle/soap_request_processing_service_policy, source=local
policy set, reference-status=enabled
    URI=oracle/ws_logging_level_policy, source=local policy set,
reference-status=enabled
    logging.level=
    URI=oracle/test_page_processing_service_policy, source=local
policy set, reference-status=enabled
    URI=oracle/wsdl_request_processing_service_policy, source=local
policy set, reference-status=enabled
    URI=oracle/wss_saml_or_username_token_service_policy,
source=local policy set, reference-status=enabled

Migrate "oracle/wss_saml_or_username_token_service_policy" (yes/no/cancel)? yes
-----
"oracle/wss_saml_or_username_token_service_policy" was migrated
successfully.-----
-----

```

For more information about the arguments for this command, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

4.10 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](#)".

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 the following sections:

- ["Attaching Policies Directly Using Fusion Middleware Control"](#)
- ["Attaching Policies Directly Using WLST"](#).

For more information about the no behavior policies, see ["No Behavior Policies"](#).

 **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 OWSM Repository"](#).

4.11 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 Using Fusion Middleware Control"](#) and ["Overriding Configuration Properties for Globally Attached Policies Using WLST"](#)
- ["Overview of Policy Configuration Overrides"](#)
- ["How the Effective Set of Policies is Calculated"](#)

4.12 Managing Endpoint Configuration Properties Using Fusion Middleware Control

After attaching a policy to an endpoint, you can view and manage the general configuration properties for that endpoint. The endpoint **Configuration** feature enables you to specify certain configuration information that you can override on a per-attachment basis, in addition to, or in lieu of setting it globally for any attachment of the policy.

Note:

For wsconfig category policies, the functionality provided by the Configuration link on the Web Service Endpoint pages is the same functionality that is provided by the **Override Policy Configuration** option provided on WSM Policy Subject Configuration pages for Directly Attached Policies.

To view and manage these endpoint configuration properties:

1. Navigate to the home page for the web service, as described in "Viewing the Web Services Summary Page for an Application" in *Administering Web Services*.
2. Select an endpoint from the Web Service Endpoints tab to open the endpoint information page.
3. Click the **Configuration** link, located near the top of the page.

The Web Service Endpoint Configuration page is displayed. From this page you can:

- Enable or disable endpoints, Restful service, WSDL, Metadata Exchange, and Endpoint Test features of the web service
 - Set the logging Level
 - Enable or disable payload schema validation
 - Configure atomic transaction flow support
 - Specify the atomic transaction version
 - Specify a maximum request size by entering a numeric value and choosing the unit type.
4. If you change any of these properties, click **Apply**.
 5. Click **Return** to go back to the Web Services summary page.

4.13 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"](#)

Because you can specify the priority of a globally or directly attached policy, as described in ["Specifying the Priority of a Policy Attachment"](#), 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 4-15](#) shows a valid configuration with a secure endpoint.

Figure 4-15 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		

Operations | **OWSM Policies** | Charts | Configuration

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

Globally Attached Policies

Policy Name	Category	Policy Set	Status	Total Violations	Authentication	Security Violator
oracle/wss_username_token_service_pol	Security	/policysets/global/ws_1	Enabled	0	0	Authorization

Directly Attached Policies

Attach/Detach

Policy Name	Category	Effective	Status	Total Violations	Authentication	Security Violations
oracle/wss10_saml_hok_token_with_mes	Security	False	Enabled	0	0	Authorization
oracle/log_policy	Management	True	Enabled	0	n/a	Confide

Using WLST, you can generate a list of endpoints and their secured status using the `listWSMPolicySubjects` WLST command. The output from these commands, when the `detail` argument is set to `true` as shown in the following example, 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.

```
wls:/jrfServer_domain/serverConfig> listWSMPolicySubjects(detail='true')
Application: /weblogic/base_domain/jaxwsejb30ws Assembly: WEB#jaxwsejb Subject: WS-
SERVICE({http://www.oracle.com/jaxws/tests/concrete}WsdConcreteService#WsdConcretePort)

    URI="oracle/mex_request_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
    URI="oracle/mtom_encode_fault_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
    URI="oracle/max_request_size_policy", category=wsconfig, policy-status=enabled;
source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
    Property name="max.request.size", value="-1"
    URI="oracle/request_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
    URI="oracle/soap_request_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
    URI="oracle/ws_logging_level_policy", category=wsconfig, policy-status=enabled;
source=local policy set; reference-status=enabled; effective=true
    Property name="logging.level", value=""
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
    URI="oracle/test_page_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
    URI="oracle/wsd_request_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
    URI="oracle/
wss11_saml_or_username_token_with_message_protection_service_policy", category=security,
policy-status=enabled; source=global policy set "domainExternal", scope="Domain(*)";
reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
```

The web service is secure in this context.

For more information about using these WLST commands, see "Viewing the Web Services in a Domain Using WLST" in *Administering Web Services*.

4.14 How the Effective Set of Policies is Calculated

OWSM 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 Attachments](#)"

OWSM policies and WebLogic web service policies cannot be attached to the same endpoint. If a Java EE endpoint has any WebLogic policies attached and you create a policy set that applies to that endpoint, the OWSM policies will be ignored when the effective policy for that endpoint is calculated.

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 scope. By default, if a subject has a policy attached at the port 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, or the same 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.

For example, the following resource scopes, in increasing order of precedence, are valid for the SOAP Web Service policy subject:

- Domain
- Application
- Application Module or Connection
- RESTful Application, Service, or Web Service Endpoint
- Port

That is, a policy attachment at the Port scope (a narrower scope) will take precedence over an attachment at the Domain scope (a broader scope). Using this 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 (at the port scope).

The valid resource scopes for each policy subject, in increasing order of precedence, are provided in "Understanding Policy Subjects" in *Understanding Oracle Web Services Manager*. For additional information about resource scopes, see "[About Defining the Type and Scope of Resources for Globally Attached Policies](#)".

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 "[Run-time Constraints in Policy Sets](#)".

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 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](#)".

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: Direct attachment. When policies with conflicting assertion categories are specified at the same scope, the directly attached policy takes precedence over the external policy attachment in the policy set.
- 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 cache management settings in the OWSM policy accessor. 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
 - Initial Cache Refresh**, default 600000 milliseconds (10 minutes)
 - Cache Refresh Time**, default 600000 milliseconds (10 minutes)

For details about tuning these properties, see "[High Availability Configuration and Cache Management Using Fusion Middleware Control](#)".

4.15 Determining the Source of Policy Attachments

WLST commands are used to determine the source of policy attachments.

To determine the source of a directly attached policy, use one of the following WLST commands with the `detail` argument set to `true`.

- `listWebServices(detail=true)`, as described in "listWebServices()" in *WLST Command Reference for Infrastructure Components*
- `listWebServiceClients(detail=true)`, as described in "listWebServiceClients()" in *WLST Command Reference for Infrastructure Components*
- `listWSMPolicySubject(detail=true)`, as described in "listWSMPolicySubjects" in *WLST Command Reference for Infrastructure Components*.

When the `detail=true` argument is specified, the output includes policy details for the services, clients, or both, including the `local.policy.reference.source` property which identifies the source of the direct attachment.

You can use this information in conjunction with the globally attached policies to determine the source of the service or client's effective policies.

Valid values for `local.policy.reference.source` are defined in [Table 4-3](#).

Table 4-3 Valid Values for local.policy.reference.source Configuration Property

Source	Description
ANNOTATION	Policy annotation specified on the web service endpoint, as described in Security and Policy Annotations for Oracle Web Services .

Table 4-3 (Cont.) Valid Values for local.policy.reference.source Configuration Property

Source	Description
IMPLIED_FEATURE	<p>Policy reference attached implicitly on web service or client endpoint.</p> <p>Note: This value applies to Oracle Infrastructure web services only.</p> <p>If not explicitly attached, the following policies are attached implicitly to the endpoint, as required:</p> <ul style="list-style-type: none"> • Configuration policy that was attached to an endpoint by default, and not explicitly by user. For more information about the configuration policies and their default values, see "Configuration Policies". • Addressing policy (oracle/wsaddr_policy), if reliable messaging is enabled, as described in "Reliable Messaging Policies", or if addressing is advertised as being required in the WSDL. • Fast Infoset policy (oracle/fastinfoset_service_policy) if advertised in the WSDL • MTOM policy (oracle/wsmtom_policy) if MTOM is supported via the deployment descriptor.
LEGACY_CONFIG	Legacy configuration defined in oracle-webservices.xml deployment descriptor (deprecated).
LEGACY_POLICYREFERENCE_PDD	Legacy policy attachment defined in oracle-webservices.xml deployment descriptor (deprecated).
LOCAL_ATTACHMENT	<p>Policy attachment using one of the following methods:</p> <ul style="list-style-type: none"> • Fusion Middleware Control, as described in "About Attaching Policies to Web Services and Clients Using Fusion Middleware Control" • WLST, as described in "About Attaching Policies to Web Services and Clients Using WLST"
PROGRAMMATIC	Programmatic policy attachment, as described in " Understanding Attaching Policies to Web Services and Clients at Design Time ".

The following provides an example of the output from the `listWebServices(detail=true)` WLST command. Note that the `local.policy.reference.source` configuration property is provided for each directly attached policy identifying the source of the attachment (shown in **bold**).

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

/base_domain/AdminServer/jaxwsejb30ws :
moduleName=jaxwsejb, moduleType=web, serviceName=CalculatorService
  CalculatorPort http://host.example.com:1234/jaxwsejb/Calculator
  URI="oracle/wss10_saml20_token_with_message_protection_service_policy",
category=security, policy-status=enabled; source=global policy set "
MyPolicySet1", scope="DOMAIN('*)"; reference-status=enabled; effective=true
  Property name="reference.priority", value="10"
  URI="oracle/mex_request_processing_service_policy",
category=wsconfig, policy-status=enabled; source=local policy set;
reference-status=enabled; effective=true
  Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
  URI="oracle/mtom_encode_fault_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
  URI="oracle/max_request_size_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source",
```

```

value="IMPLIED_FEATURE"
    Property name="max.request.size", value="-1"
    URI="oracle/request_processing_service_policy", category=wsconfig,
    policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
    URI="oracle/soap_request_processing_service_policy", category=wsconfig,
    policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
    URI="oracle/ws_logging_level_policy", category=wsconfig,
    policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="logging.level", value=""
    Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
    URI="oracle/test_page_processing_service_policy", category=wsconfig,
    policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
    URI="oracle/wsdl_request_processing_service_policy", category=wsconfig,
    policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
    URI="oracle/http_saml20_token_bearer_service_policy", category=security,
    policy-status=enabled; source=local policy set; reference-status=enabled; reference-
    status=enabled; effective=false
    Property name="local.policy.reference.source", value="LOCAL
ATTACHMENT"

```

The policy subject is secure in this context.

5

Overriding Policy Configuration Properties

This topic describes the policy configuration override feature which allows you to specify certain web service configuration information in a policy that you can override on a per-attachment 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.

It includes the following topics:

- [Overview of Policy Configuration Overrides](#)
- [Scope of Predefined Configuration Properties](#)
- [About Overriding Client Policy Configuration Properties at Design Time](#)
- [About Overriding Policy Configuration Properties Using Fusion Middleware Control](#)
- [About Overriding Policy Configuration Properties Using WLST](#)
- [About Configuring User-Defined Properties for Web Service and Client Policies Using Fusion Middleware Control](#)

5.1 Overview of Policy Configuration Overrides

Values for server-side configuration properties in a predefined or custom web service policy can be used each time you attach the policy to a web service or overridden on a per-attachment basis. For web service clients, configuration can be overridden on a per-client basis. One of the possible uses of overrides is to limit the number of policies you have to maintain: rather than creating multiple policies with slightly varied configurations, you can use the same generic policy and override specific values to meet your requirements.

 **Note:**

Unless specified otherwise, the procedures in this chapter apply to Oracle Infrastructure web services and RESTful web services only.

Configuration properties that you can override are of two types:

- **Predefined policy configuration properties**—The configuration properties included with the predefined service policies allow you to override certain domain-wide configuration settings, such as the CSF key used for storing the signature-key password. For predefined client policies, you can override a configuration property on a per-client basis or set it globally for any attachment of the policy.

The configuration properties that you can override in a predefined policy are inherited from the assertion templates that are included in the policy. To determine the configuration properties associated with each policy, see [Oracle Web Services Manager Predefined Policies](#). An alphabetized list of the overrideable properties is provided in "[Assertion Template Configuration Properties for Oracle Web Services](#)". Note that you cannot override a property of type "constant".

- 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 property to a cloned predefined policy, or to a custom policy. For more information about creating and configuring user-defined policy configuration properties, see "[About Configuring User-Defined Properties for Web Service and Client Policies Using Fusion Middleware Control](#)".

 **Note:**

The predefined policies are read-only and cannot be edited. You can, however, create new policies using the predefined policies as a base. For information about creating a new policy, see "[Creating and Editing Web Service Policies](#)". Once you have created the new policy, you can edit the policy and set the configuration properties as desired.

When attaching OWSM 14c predefined policies, if you specify a value of blank (" ") in the Value field, the default value will be in effect. If you have imported 11g policies or any custom policies, ensure that the policy has a valid value in the Default field to achieve the same effect; otherwise, the specified value will be picked up.

5.2 Scope of Predefined Configuration Properties

The scope for the server-side configuration property value is limited to the specific policy attachment. 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.

The scope for a client-side configuration property value is the client. There can be multiple policies that are attached to the same client that use the same property. 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 clone the `oracle/wss_http_token_client_policy` and set the value for the configuration property in the cloned version and the new value can apply to every instance. You also have the option to override the configuration property on a per-client basis when you attach the policy.

 **Note:**

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.

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.

5.3 About Overriding Client Policy Configuration Properties at Design Time

You can override client policy configuration properties for an OWSM security policy programmatically at design time.



Note:

The procedures in this section apply to Java EE, RESTful, and Oracle Infrastructure web services.

Following methods are used to override client policy configuration:

- [Java EE Web Services](#).
- [RESTful Web Services](#).
- [Understanding Oracle Infrastructure Web Services](#).

5.3.1 Java EE Web Services

Client Policy Configuration can be overridden by using Java EE Web Services.

- `JAX-WS RequestContext`, as shown in [xREFTBD](#).
- `weblogic.wsee.jws.jaxws.owsm.Property` annotation when attaching an OWSM security policy to Java EE web services and clients, as described in [Attaching Policies to Java EE Web Services and Clients Using Annotations](#).
- Using JDeveloper, as described in "Attaching Policies" in *Developing Applications with Oracle JDeveloper*.

5.3.2 RESTful Web Services

You can override client policy configuration properties for an OWSM security policy by using RESTful Web Services method.

- `oracle.wsm.metadata.annotation.Property` annotation when attaching an OWSM security policy to RESTful web services, as described in [Attaching Policies to RESTful Web Services Using Annotations](#).
- `oracle.wsm.metadata.feature.PropertyFeature` annotation when attaching an OWSM security policy to RESTful web service clients, as described in [Attaching Policies to RESTful Web Service Clients Using Feature Classes](#). See also [#unique_333/unique_333_Connect_42_CEGIDDAE](#).
- Using JDeveloper, as described in "Attaching Policies" in *Developing Applications with Oracle JDeveloper*.

5.3.3 Understanding Oracle Infrastructure Web Services

OWSM security policy can be attached to Oracle Infrastructure Web services using annotations or JDeveloper.

- `oracle.wsm.metadata.annotation.Property` annotation when attaching an OWSM security policy to Oracle Infrastructure web services, as described in [Attaching Policies to Oracle Infrastructure Web Services Using Annotations](#).
- `oracle.wsm.metadata.feature.PropertyFeature` annotation when attaching an OWSM security policy to Oracle Infrastructure web service clients, as described in [Attaching Policies to Oracle Infrastructure Web Service Clients Using Feature Classes](#).
- Using JDeveloper, as described in "Attaching Policies" in *Developing Applications with Oracle JDeveloper*.

To clear a client policy configuration property, set it to the empty string. Before clearing it, consider the other policies that might be using the same property. For web service clients, configuration properties are client-specific; there may be multiple policies attached to the same client that use the same property.

For more information, refer to the following sections:

- [Client Policy Configuration Properties That Can Be Overridden at Design Time](#)
- [Example for Overriding the Client Policy Configuration Properties for Keystore, Username, and Password Using RequestContext](#)
- [Example for Overriding the RESTful Web Service Client Policy Configuration Properties for the Username and Password](#)

5.3.3.1 Client Policy Configuration Properties That Can Be Overridden at Design Time

You can override Client Policy Configuration Properties at design time.

[Table 5-1](#) lists the client-side configuration properties you can override programmatically and the policies to which each property applies.



Note:

For JSE clients, you need to configure the `jps-config-jse.xml` in OPSS for access to the csf keys. For more information about configuring the `jps-config-jse.xml` file, see "Using OPSS in Java SE Applications" in *Securing Applications with Oracle Platform Security Services*.

Table 5-1 Client Policy Configuration Properties That Can Be Overridden at Design Time

Property	Description
<code>BindingProvider.USERNAME_PROPERTY</code> (<code>javax.xml.ws.security.auth.username</code>)	User name for authentication.
<code>BindingProvider.PASSWORD_PROPERTY</code> (<code>javax.xml.ws.security.auth.password</code>)	Password for authentication.
<code>oracle.wsm.security.util.SecurityConstants.ClientConstants.ATTESTING_MAPPING_ATTRIBUTE</code>	Mapping attribute used to represent the attesting entity. Only the DN is currently supported. This attribute is applicable only to sender vouches message protection use cases. It is not applicable to SAML over SSL policies.

Table 5-1 (Cont.) Client Policy Configuration Properties That Can Be Overridden at Design Time

Property	Description
<code>oracle.wsm.security.util.SecurityConstants.ClientConstants.CALLER_PRINCIPAL_NAME</code>	Principal name of the client, as generated using the <code>ktpass</code> command and mapped to the username, for which the kerberos token should be generated. Use the following format: <code><username>@<REALM NAME></code> . Note: <code>keytab.location</code> and <code>caller.principal.name</code> are required for propagating client identity for Java EE applications.
<code>oracle.wsm.security.util.SecurityConstants.ClientConstants.ON_BEHALF_OF</code>	Flag that specifies whether the request is on behalf of another entity. When set to <code>true</code> (the default) and <code>sts.auth.on.behalf.of.csf.key</code> is configured, then it will be given preference and the identity established using that CSF key will be sent as <code>onBehalfOf</code> token. Otherwise, if the subject is already established, then the username from the subject will be sent as <code>onBehalfOf</code> token. If <code>sts.auth.on.behalf.of.csf.key</code> is not set and the subject does not exist, <code>on.behalf.of</code> is treated as a token exchange for the requestor and not for another entity. It is not included in an <code>onBehalfOf</code> element in the request.
<code>oracle.wsm.security.util.SecurityConstants.ClientConstants.SAML_AUDIENCE_URI</code>	Relying party, as a comma-separated URI. This property accepts wildcards. For more information, see " saml.audience.uri ".
<code>oracle.wsm.security.util.SecurityConstants.ClientConstants.STS_KEY_STORE_RECIPIENT_ALIAS</code>	Public key alias of the STS.
<code>oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_INCLUDE_USER_ROLES</code>	User roles in a SAML assertion.
<code>oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_ISSUED_TOKEN_LIFETIME</code>	The time in milliseconds for OWSM to request as the token lifetime when obtaining an issued token from a security token service (STS). See " issued.token.lifetime " for more information.
<code>oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_PASSWORD_PROPERTY</code>	Password for the RESTful web service client. Note: This property is valid for RESTful web service clients only.
<code>oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_SAML_ASSERTION_FILE_NAME</code>	File containing the assertions for SAML HOK policies.
<code>oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_STS_AUTH_ON_BEHALF_OF_CSF_KEY</code>	On behalf of entity. If present, it will be given preference over Subject (if it exists).
<code>oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_STS_AUTH_USER_CSF_KEY</code>	Username/password to authenticate to the STS. If <code>policy-reference-uri</code> in the STS configuration policy points to a username-based policy, then you configure the <code>sts.auth.user.csf.key</code> property to specify a username/password to authenticate to the STS.

Table 5-1 (Cont.) Client Policy Configuration Properties That Can Be Overridden at Design Time

Property	Description
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_STS_AUTH_X509_CSF_KEY	X509 certificate for authenticating to the STS. If the <code>policy-reference-uri</code> in the STS configuration policy points to an x509-based policy, then configure the <code>sts.auth.x509.csf.key</code> property to specify the X509 certificate for authenticating to the STS.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_SUBJECT_PRECEDENCE	Flag that specifies whether to use the OWSM subject. Set to <code>false</code> to use a client-specified username.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_USE_RNAME_PROPERTY	Username for the RESTful web service client. Note: This property is valid for RESTful web service clients only.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_CSF_KEY	Username and password corresponding to the <code>csf-key</code> specified in the credential store, if the credential store is available to the client. Alternatively, you can set the username and password explicitly. See the example later in this section.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_ENC_KEY_ALIAS	Alias of the key within the keystore that will be used to decrypt the response from the service. This property overrides any statically configured value. This property is not used in WSS11 policies. Type: <code>java.lang.String</code>
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_ENC_KEY_PASSWORD	Password for the key within the keystore that will be used for decryption. This property overrides any statically configured value. This property is not used in WSS11 policies. Type: <code>java.lang.String</code>
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KERBEROS_SERVICE_PRINCIPAL	Service principal name to use when trying to access a service that is protected using the Kerberos mechanism. This property overrides any static configuration value. Type: <code>java.lang.String</code>
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_LOCATION	Location of the keystore file. For KSS, this is the KSS URI. This property overrides any statically configured value. Type: <code>java.lang.String</code>
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_PASSWORD	Password of the keystore file. This property overrides any statically configured value. Type: <code>java.lang.String</code>
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_TYPE	Type of keystore file. This property overrides any statically configured value. Valid values include: JKS and KSS (default).
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_RECIPIENT_KEY_ALIAS	Alias for the recipient's public key that is used to encrypt type outbound message. This property overrides any static configuration value. Type: <code>java.lang.String</code>

Table 5-1 (Cont.) Client Policy Configuration Properties That Can Be Overridden at Design Time

Property	Description
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_SAML_ISSUER_NAME	SAML issuer name to use when trying access a service that is protected using SAML mechanism. This property overrides any static configuration value. Type: java.lang.String
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_SIG_KEY_ALIAS	Alias of the key within the keystore that is used for digital signatures. This property overrides any statically configured value. For WSS11 policies, this property is used for mutual authentication only. Type: java.lang.String
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_SIG_KEY_PASSWORD	Password for the alias of the key within the keystore that is used for digital signatures. This property overrides any statically configured value. For WSS11 policies, this property is used for mutual authentication only. Type: java.lang.String
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_WSDL_URI	The endpoint URI of an STS WSDL, used to obtain STS information and invoke the STS for token exchange.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_PORT_URI	The actual endpoint URI of the STS port. For example: http://host:port/context-root/service1.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_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).
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_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.

5.3.3.2 Example for Overriding the Client Policy Configuration Properties for Keystore, Username, and Password Using RequestContext

The `RequestContext` command override the client policy configuration properties for keystore, username, and 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();
}
}
}

```

WIP

The following example shows `c:/dat/client-pdd.xml` referenced in the previous example:

```

! -- 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>

```

5.3.3.3 Example for Overriding the RESTful Web Service Client Policy Configuration Properties for the Username and Password

The following example shows an example of how to override the RESTful web service client policy configuration properties for the username and password.

```

package example;
import oracle.wsm.security.utils.SecurityConstants;
import com.sun.jersey.api.client.config.ClientConfig;
import com.sun.jersey.api.client.config.DefaultClientConfig;
import oracle.wsm.metadata.feature.PolicyReferenceFeature;
import oracle.wsm.metadata.feature.AbstractPolicyFeature;
import oracle.wsm.metadata.feature.PolicySetFeature;
import oracle.wsm.metadata.feature.PropertyFeature;
...
public class MyRESTfulClient {
    public static void main(String[] args) {
    ...
        PropertyFeature uname = new

```

```
PropertyFeature (SecurityConstants.ClientConstants.WSM_USERNAME_PROPERTY, "yourusername");
    PropertyFeature pwd = new

PropertyFeature (SecurityConstants.ClientConstants.WSM_PASSWORD_PROPERTY, "yourpassword");
    PropertyFeature[] propFeatures = new PropertyFeature[] { uname, pwd };
    PolicyReferenceFeature clientPRF = new
        PolicyReferenceFeature("oracle/wss_http_token_client_policy");
    ClientConfig cc = new DefaultClientConfig();
    Map<String, Object> properties = cc.getProperties();
    properties.put (AbstractPolicyFeature.ABSTRACT_POLICY_FEATURE, new
        PolicySetFeature(clientPRF, propFeatures));
    ...
}
}
```

5.4 About Overriding Policy Configuration Properties Using Fusion Middleware Control

Web services configuration can be overridden at the domain level, the web service application port level (direct attachments), and at the client application level.

Topics:

- [Overriding Configuration Properties at the Domain Level \(Defining the Default Value\)](#)
- [Overriding Configuration Properties for Directly Attached Service Policies Using Fusion Middleware Control](#)
- [Overriding Configuration Properties at the Web Service Client Application Level Using Fusion Middleware Control](#)
- [Overriding Configuration Properties for Globally Attached Policies Using Fusion Middleware Control](#)

5.4.1 Overriding Configuration Properties at the Domain Level (Defining the Default Value)

To override configuration properties at the domain level, you can change the default value of a configuration override property in a policy. When you attach the policy to a web service or client, any web service to which the policy is attached can use these values, or you can override the value when you attach the policy.

For example, you may want to use domain level configuration overrides for keystore configuration and authorization settings:

- The predefined OWSM message protection policies define a set of server-side override properties such as `keystore.sig.csf.key` and `keystore.enc.csf.key`. By default, these properties have a blank value. If you set (or then override) any of the server-side configuration properties, then 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 "[Overview of Configuring Keystores for Message Protection](#)".

If you do not set these properties and leave the default values, then the values you configure as part of setting up the keystore for message protection are used instead, as described in "[Overview of Configuring Keystores for Message Protection](#)".

- The predefined `oracle/binding_permission_authorization_policy` defines a set of server-side override properties: `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 match patterns you configure as described in "[Determining Authorization Permissions](#)".

 **Note:**

The predefined policies are read-only and cannot be edited. You can, however, create new policies using the predefined policies as a base. For information about creating a new policy, see "[Creating and Editing Web Service Policies](#)". Once you have created the new policy, you can edit the policy and set the configuration properties as desired.

Perform the following steps to set a value for a configuration property in a policy:

1. Navigate to the Web Services Policies page, as described in "[Navigating to the WSM Policies Page](#)".
2. From the WSM Policies page, select the cloned policy for which you want to set the default value and click **Open**.
3. Select the **Assertions** tab, then click **Configuration**.
4. In the Configuration page, enter the desired value in the **Value** field for the property.
5. Click **OK**.
6. Click **Validate** to validate the policy.
7. Click **Save**.

5.4.2 Overriding Configuration Properties for Directly Attached Service Policies Using Fusion Middleware Control

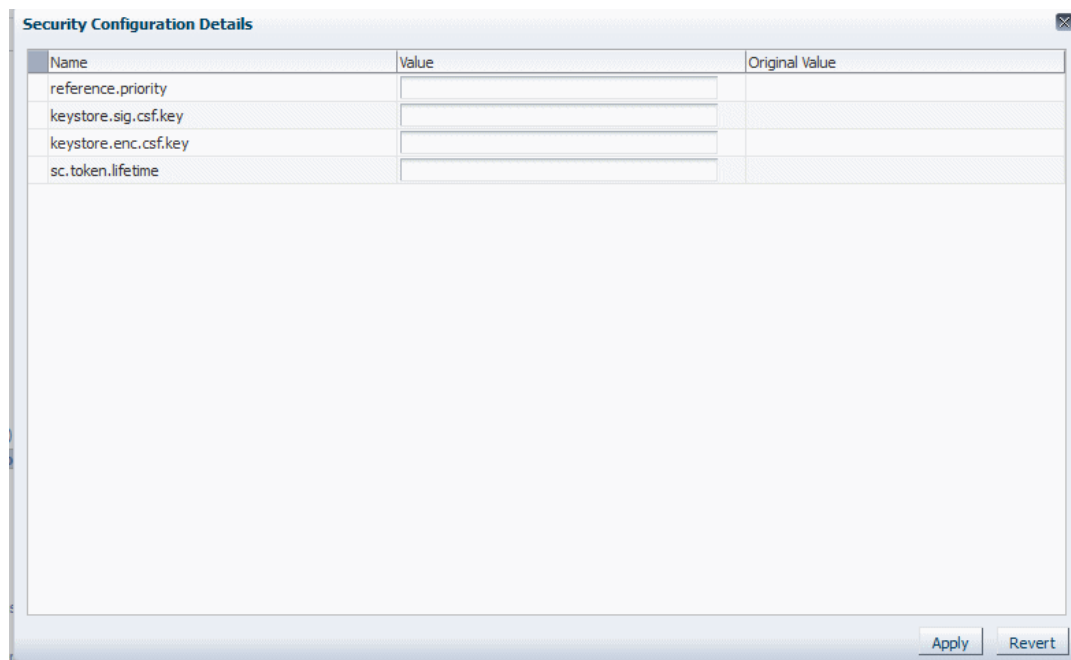
To override configuration properties for directly attached policies, attach the policy to the endpoint in the application, and then override the value for the desired property in the attached policy. Note that you do not have to clone a predefined policy to configure policy overrides at the application level because you are not changing the policy.

Perform the following steps to override a configuration property for a directly attached policy using Fusion Middleware Control:

1. Attach the policy to the endpoint as described in "[Attaching Policies Directly Using Fusion Middleware Control](#)".
2. In the Directly Attached Policies table, select the policy that contains the property to be overwritten and click **Override Policy Configuration**.
3. Select **Override Policy Configuration**.

The **Security Configuration Details** window is displayed, as shown in [Figure 5-1](#). This figure shows the overridable properties for the `oracle/wss10_message_protection_service_policy`.

Figure 5-1 Overriding a Policy Configuration Property



4. 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 "[Overview of Configuring Keystores for Message Protection](#)".

5.4.3 Overriding Configuration Properties at the Web Service Client Application Level Using Fusion Middleware Control

You can override configuration properties at the web service client application level for Java EE and Oracle Infrastructure web service.



Note:

The procedures in this section apply to Java EE and Oracle Infrastructure web service clients only.

Perform the following steps to to override a client configuration property using Fusion Middleware Control:

1. Attach a policy to a web service client, as described in "[Attaching Policies Directly to Web Service Clients Using Fusion Middleware Control](#)".
2. In the Directly Attached Policies table on the client endpoint page, select the policy and click **Override Policy Configuration**.
3. Enter the override value in the **Value** field for the property and click **Apply**.
The property is overridden on a per-attachment basis.

5.4.4 Overriding Configuration Properties for Globally Attached Policies Using Fusion Middleware Control

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 Fusion Middleware Control. Because global policy attachments can be scoped at a higher level than direct policy attachments, such as application or domain level, configuration overrides configured in the policy set also apply at the higher scope.



Note:

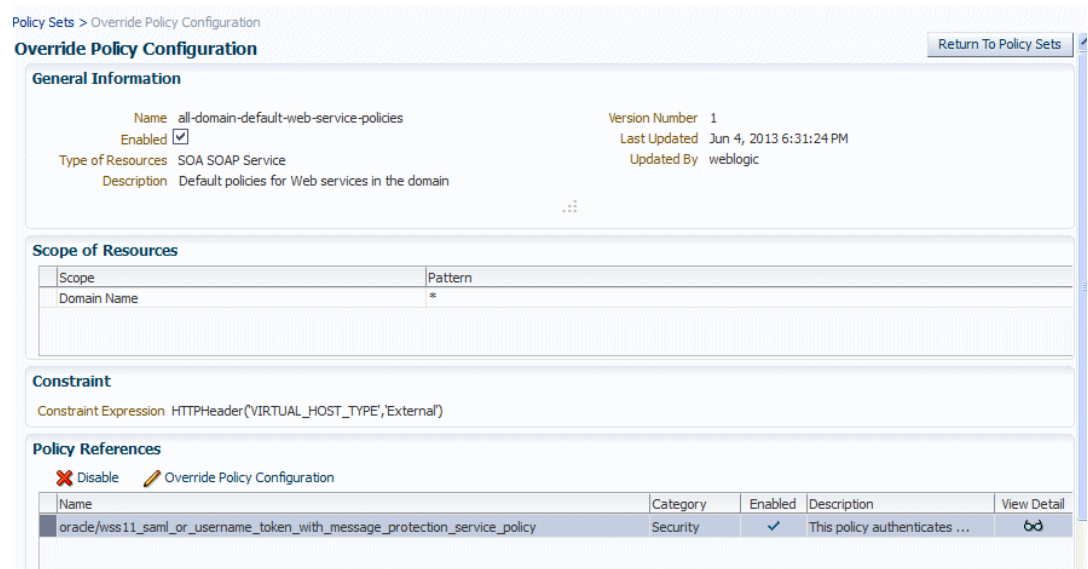
The procedure in this section applies to Java EE, RESTful, and Oracle Infrastructure web services.

Perform the following steps to override a configuration property in a policy referenced in a policy set:

1. Go to the WSM Policy Set Summary page as described in "[Navigating to the WSM Policy Set Summary Page Using Fusion Middleware Control](#)".
2. From the WSM Policy Set Summary page, select the policy set containing the policy for which you want to configure overrides.
3. Select **Override Policy Configuration**.

The Override Policy Configuration page is displayed, as shown in [Figure 5-2](#).

Figure 5-2 Policy Set Override Policy Configuration Page



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.
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.

5.5 About Overriding Policy Configuration Properties Using WLST

Web services configuration can be overridden at the web service application level, at the client application level, and for globally attached policies.

Topics:

- [Overriding Configuration Properties for Directly Attached Service Policies Using WLST](#)
- [Overriding Configuration Properties at the Web Service Client Application Using WLST](#)
- [Overriding Configuration Properties for Globally Attached Policies Using WLST](#)

5.5.1 Overriding Configuration Properties for Directly Attached Service Policies Using WLST

When you attach a policy that has an overridable property, you can override the existing value using the `setWSMPolicyOverride` command.



Note:

The procedure in this section applies to Oracle Infrastructure and RESTful web services only.

Perform the following steps to override configuration properties for directly attached service policies using WLST:

1. Attach the policy to the service or client as described in "[Attaching Policies Directly Using WLST](#)".
2. Within a session, use the `setWSMPolicyOverride` command to override policy properties.

```
setWSMPolicyOverride(policyURI,property, value)
```

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>setWSMPolicyOverride("oracle/  
wss10_message_protection_service_policy","keystore.sig.csf.key","sigkey")
```

The configuration override property `"keystore.sig.csf.key"` having value `"sigkey"` has been added to the reference to policy with URI `"oracle/wss10_message_protection_service_policy"`.



Note:

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.

The `local.policy.reference.source` property is for informational purposes only, to identify the source of the direct policy attachment, and should not be overridden. For more information, see "[Determining the Source of Policy Attachments](#)".

For more information about this WLST command and its arguments, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

5.5.2 Overriding Configuration Properties at the Web Service Client Application Using WLST

When you attach a client policy that has an overridable property, you can override the existing value using the `setWebServiceClientStubProperty` or `setWebServiceClientStubProperties` commands.

 **Note:**

This procedure applies to Oracle Infrastructure web service clients only.

Perform the following steps to override a client configuration property using WLST:

1. Attach the policy to the web service client, as described in "[About Attaching Policies Directly to RESTful and Oracle Infrastructure Web Services and Clients Using WLST](#)".
2. Use the `setWebServiceClientStubProperty` or `setWebServiceClientStubProperties` command to override policy properties.

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

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

For example, to set or override multiple properties:

```
wls:soainfra/serverConfig>  
setWebServiceClientStubProperties('/base_domain/soa_server1/adf_dc_to_bc',  
'ADF_BC', 'wsconn', 'AppModuleService', 'AppModuleServiceSoapHttpPort',  
[('csf-key', 'HCM_APPID'), ('keystore.recipient.alias', 'orakey')])
```

 **Note:**

You should not override the `local.policy.reference.source` property. This property is for informational purposes only, to identify the source of the direct policy attachment. For more information, see "[Determining the Source of Policy Attachments](#)".

For more information about this WLST command and its arguments, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

5.5.3 Overriding Configuration Properties for Globally Attached Policies Using WLST

You can specify a configuration override in a policy referenced in a policy set using the `setWSMPolicyOverride` command. This command can be used only during the creation or modification of a policy set within the context of a session.

Note:

- The procedure in this section applies to Java EE, RESTful, and Oracle Infrastructure web services.
- You can also set a configuration override scoped to a policy set using the `setWSMPolicySetOverride` command. For more information, see "setWSMPolicySetOverride" in *WLST Command Reference for Infrastructure Components*

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 session while creating a new policy set or creating a policy set from an existing policy set.

1. Begin a session using the `beginWSMSession` command.

For example:

```
wls:/jrfserver_domain/serverConfig> beginWSMSession()
```

```
Session started for modification.
```

2. Use the `selectWSMPolicySet` command to select an existing policy set to edit.

```
selectWSMPolicySet (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> selectWSMPolicySet ('default-domain-ws-domain_gpa')
```

```
The policy set is ready for modification in the session.
```

3. Optionally, view the configuration of the policy set using the `displayWSMPolicySet` command.

For example:

```
wls:/jrfserver_domain/serverConfig>displayWSMPolicySet()
```

```
Policy Set Details:
```

```
-----
```

```
Display Name : default-domain-ws-domain_gpa
```

```
Type of Resources: SOAP Web Service
```

```
Scope of Resources: DOMAIN('base_domain')
```

```
Description: Global policy attachments for Web Service Endpoint resources.
```

```
Enabled: true
```

```
Policy Reference: URI=oracle/
```

```
wss11_saml_or_username_token_with_message_protection_service_policy,
```

```
category=security, enabled=true, index=1
        URI=oracle/log_policy, category=management, enabled=false,
index=2
```

4. Specify the configuration override for the policy reference using the `setWSMPolicyOverride` command.

For example, to specify a configuration override for the `reference.priority` property for `oracle/wss11_saml_or_username_token_with_message_protection_service_policy`, enter the following command:

```
wls:/jrfserver_domain/serverConfig>setWSMPolicyOverride('oracle/
wss11_saml_or_username_token_with_message_protection_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/wss11_saml_or_username_token_with_message_protection_service_policy".

5. Optionally, view the configuration of the policy set.

For example:

```
wls:/jrfserver_domain/serverConfig>displayWSMPolicySet()
```

Policy Set Details:

```
Display Name : default-domain-ws-domain_gpa
Type of Resources: SOAP Web Service
Scope of Resources: DOMAIN('base_domain')
Description: Global policy attachments for Web Service Endpoint resources.
Enabled: true
Policy Reference: URI=oracle/
wss11_saml_or_username_token_with_message_protection_service_policy,
category=security, enabled=true, index=1
reference.priority=1
```

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 `validateWSMPolicySet` command.

For example:

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

The global policy set default-domain-ws-domain_gpa is valid.

7. To write the contents of the current session to the repository, use the `commitWSMSession` command.

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

The policy set default-domain-ws-domain_gpa is valid.
Updating policy set default-domain-ws-domain_gpa in repository.

Session committed successfully.

5.6 About Configuring User-Defined Properties for Web Service and Client Policies Using Fusion Middleware Control

You can add one or more user-defined server- or client-side properties that have meaning in your environment to a cloned copy of a predefined policy, or to a custom policy. Then, you can either use the user-defined property as-is, or override it when you attach the policy.



Note:

The procedures described in this section apply to Oracle Infrastructure and Restful web services only.

When you either use the user-defined property or override it when you attach the policy, 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 cloned version of 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](#)
- [Adding a User-Defined Configuration Property](#)
- [Editing a User-Defined Configuration Property](#)
- [Deleting a User-Defined Configuration Property](#)
- [Overriding the User-Defined Configuration Properties](#)

5.6.1 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 can 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 can 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.

5.6.2 Adding a User-Defined Configuration Property

You can edit a cloned copy of a predefined policy, or a custom policy, to add a user-defined configuration property.

Perform the following steps to add a user-defined configuration property:

1. Navigate to the WSM Policies page, as described in "[Navigating to the WSM Policies Page](#)".
2. From the WSM Policies page, select the policy for which you want to add the property and click **Open**.
3. Select the **Assertions** tab, and then click **Configuration**.
4. In the Configuration page, click **Add** to add the new property.
5. In the table, provide a name and a value for the new property. The **Name** field is required and must be unique for the policy.
6. From the **Type** menu, select **constant**, **optional**, or **required**. You can subsequently override only properties of type optional and required.
7. Click **OK**.
8. Click **Validate** to validate the policy.
9. Click **Save**.

5.6.3 Editing a User-Defined Configuration Property

User-defined configuration properties can be edited and validated.

Perform the following steps to edit a user-defined configuration property:

1. Navigate to the WSM Policies page, as described in "[Navigating to the WSM Policies Page](#)".
2. From the WSM Policies page, select the policy for which you want to edit the property and click **Open**.
3. Select the **Assertions** tab, and then click **Configuration**.
4. In the Configuration page, edit the property as required and click **OK**.
5. Click **Validate** to validate the policy.
6. Click **Save**.

5.6.4 Deleting a User-Defined Configuration Property

You can delete a user-defined configuration property if you no longer need it.

Perform the following steps to delete a user-defined configuration property:

1. Navigate to the WSM Policies page, as described in "[Navigating to the WSM Policies Page](#)".
2. From the WSM Policies page, select the policy for which you want to edit the property and click **Open**.
3. Select the **Assertions** tab, and then click **Configuration**.
4. In the Configuration page, select the user-defined property to be deleted and click **Delete**.

5. Click **OK**.
6. Click **Validate** to validate the policy.
7. Click **Save**.

5.6.5 Overriding the User-Defined Configuration Properties

You can override a user-defined configuration property using the same methods that you use for predefined or custom policies.

For more information, see the following topics:

- ["About Overriding Policy Configuration Properties Using Fusion Middleware Control "](#)
- ["About Overriding Policy Configuration Properties Using WLST "](#)

6

Managing Web Service Policies with Fusion Middleware Control

For information about web services policies and how Oracle Web Services Manager (OWSM) uses policies to manage Quality of Service (QoS) for web services, see [Overview of OWSM Policy Framework](#) in *Understanding Oracle Web Services Manager*. The following sections describe managing web service policies with fusion middleware control:

- [Overview of Web Services Policy Management](#)
- [Managing Web Service Policies](#)
- [Validating Web Service Policies](#)
- [Managing Policy Assertion Templates](#)
- [Managing Policies and Assertions](#)
- [About Advertising Policy Assertions](#)
- [About Advertising WS-Policy and WS-SecurityPolicy Versions](#)

6.1 Overview of Web Services Policy Management

In the 14c release, the predefined documents delivered with OWSM, including policies and assertion templates, are read-only. If this is a new installation, then all of the documents that are installed with OWSM will be read-only. To modify a predefined policy or assertion template, you will need to clone it and then make the desired modifications to the cloned version.

If you are installing into an existing OWSM environment, or if you are upgrading from an older release, any predefined documents that have not been customized for your environment are replaced with read-only versions, and new predefined read-only documents are added. Note, however, that any existing predefined documents that you have customized, and user-created custom policies in the repository are not overwritten.

 **Note:**

To ensure that you always get all of the latest policies, Oracle recommends that you clone any predefined documents that you have modified and migrate any policy attachments. For details, see "[Upgrading the OWSM Repository](#)".

6.2 Managing Web Service Policies

From **Managing Web Service Policies** page you can search for specific policies or types of policies, view policies, create new policies, edit custom policies, delete custom policies, and import and export custom policies to or from the OWSM repository.

Topics:

- [Navigating to the WSM Policies Page](#)

- Searching for Policies in the WSM Policies Page
- Viewing the Details of a Web Service Policy
- Creating and Editing Web Service Policies
- Using Local Optimization with OWSM Policies (SOA Composites)
- Generating Client Policies from a WSDL
- Adding Assertions to a Policy
- Adding an OR Group to a Policy
- Importing Web Service Policies
- Exporting Web Service Policies
- Versioning Web Service Policies
- Deleting a Web Service Policy

6.2.1 Navigating to the WSM Policies Page

Use the **WSM Policies** page to manage the web service policies. From this page you can search for specific policies or types of policies, view policies, create new policies, edit custom policies, delete custom policies, and import and export policies to or from the OWSM repository.

Figure 6-1 WSM Policies Page

The screenshot displays the 'WSM Policies' page. At the top, there is a search section with a 'Name' dropdown set to 'Contains' and a 'Category' dropdown set to 'All'. Below the search section is a toolbar with various actions: Create, Create Like, Open, Delete, Export, Import, Generate Client Policy, and Detach. The main area contains a table with the following columns: Name, Category, Status, Attachment, and Description. The table lists several policies, including 'oracle/addressing_routing_client_policy', 'oracle/async_web_service_policy', 'oracle/atomic_transaction_policy', and several 'oracle/binding_...' policies. The status for all listed policies is '✓' (green checkmark) and the attachment count is '0'. The description for the first policy is 'This policy facilitates setting client-side response r...'. The bottom of the table indicates 'Showing 153 out of 153 Rows'.

Name	Category	Status	Attachment	Description
oracle/addressing_routing_client_policy	Configuration	✓	0	This policy facilitates setting client-side response r...
oracle/async_web_service_policy	Configuration	✓	0	This policy facilitates enabling and configuring JRF...
oracle/atomic_transaction_policy	Atomic Transactions	✓	0	This policy facilitates enabling WS-AT Atomic Trans...
oracle/binding_authorization_denyall_policy	Security	✓	0	This policy is a special case of simple role based au...
oracle/binding_authorization_permitall_policy	Security	✓	0	This policy is a special case of simple role based au...
oracle/binding_oes_authorization_policy	Security	✓	0	This policy does user authorization based on polic...
oracle/binding_oes_masking_policy	Security	✓	0	This policy does response masking based on policy...
oracle/binding_permission_authorization_policy	Security	✓	0	This policy is a special case of simple Permission ba...

6.2.2 Searching for Policies in the WSM Policies Page

Use the **WSM Policies** page to search for policies using the advanced search feature, the Query by Example filter, or a combination of the two to refine the search.

Details of searching for policies is provided in the following sections:

- [Using Advanced Search](#)
- [Using the Query by Example Filter](#)

6.2.2.1 Using Advanced Search

In the **WSM Policies** page, you can reduce the number of policies that are returned by specifying the appropriate search criteria. To do so, perform the following steps:

1. In the Search pane, specify the criteria to use in the search:
 - In the **Name** field, enter a policy name or part of a policy name and select the operator to use to refine the search. Available operators are Starts with, Ends with, Equals, and Contains. For example, to search for message protection policies only, select the **Contains** operator, and enter `message` in the **Name** field.

You can use percent `%` as a wildcard, any place in the name. Asterisk `*` is not recognized as a wildcard and is treated as plain text. Searches are case-insensitive.

- In the **Category** field, select the desired category.

Alternatively, you can select one of the previously saved searches from the **Saved Search** drop-down menu. The search parameters automatically populate the search fields. If the **Run automatically** option is specified for the saved search, it runs automatically and the results are displayed in the Policies table.

2. Optionally, refine the search using the Query By Example filter, as described in "[Using the Query by Example Filter](#)". Note that when you combine the two search types, the data entered into the Query By Example fields is appended using the AND operator to the data specified in the Search fields.

3. Click **Search**.

The Policies table is refreshed to include only those policies that match the specified search criteria. For example, using the example specified above for message protection policies only, if you did not refine the search using Query by Example, all message protection policies are shown in the list. If you used Query by Example to refine the search for client policies only, the list displayed includes only message protection client policies.

4. Optionally, click **Save...** to save the search criteria in the repository. Note that only the values specified in the Advanced Search fields are saved; the values specified in the Query By Example fields are not included in the saved searches.

In the Create Saved Search window, enter a name for the search in the Name field. To use this saved search as the default selection for future searches, select **Set as default**. To execute the search automatically when it is selected, select **Run automatically**. Click **OK**.

To modify previously saved searches, click **Personalize...** from the **Saved Search** drop-down menu. In the Personalize Saved Searches window, select the saved search from the drop-down menu, edit as required and click **Apply**. To delete a saved search click **Delete**. To duplicate a search, click **Duplicate**. Modify the duplicate as desired and click **Apply**. When you are finished editing all of the searches, click **OK**.

6.2.2.2 Using the Query by Example Filter

The Query by Example filter in the WSM Policies page allows you to query a specific field and filter the results displayed in the table quickly and easily.

1. If the search fields are not displayed at the top of the Policies table, click the **Query by Example** icon. A search field is displayed above the Name, Category, and Status columns.

2. Enter the search criteria in the field above the column in which you want to search. The value entered is interpreted as a "contains" expression. That is, the value is wrapped in `%value%`, and will fetch all results that contain the value specified for that column. For example, to search for client policies only, enter `client` in the search field above the **Name** column.
3. If you are using the Query by Example filter separately (not in conjunction with the advanced search fields), press **Enter**.

The list of policies displayed in the table is filtered to display only the results that match the search criteria. Using the example specified in previous step, only client policies are displayed.

 **Note:**

The Query by Example search fields can be used in conjunction with the advanced search fields to further refine the search results, as described in "[Using Advanced Search](#)". When used together, the data entered into the Query By Example fields is appended using the AND operator to the data specified in the Search fields. You must use the **Search** button to get the combined results.

You need to manually clear the Query by Example search fields when you have completed the search.

6.2.3 Viewing the Details of a Web Service Policy

Use the **WSM Policies** page to view the details of a web service policy.

Predefined policies from Oracle are read-only and cannot be modified. These policies are displayed in read-only mode. User-created policies are not read-only, and can be edited as described in "[Editing a Web Service Policy](#)".

Perform the following steps to view the details of a web service policy:

1. Navigate to the WSM Policies page as described in "[Navigating to the WSM Policies Page](#)".

Optionally, refine the list of policies displayed using Search, as described in "[Searching for Policies in the WSM Policies Page](#)".

2. Select the policy to be viewed from the list of policies and click **Open**. Alternatively, select **Actions** and then **Open**.

[Figure 6-2](#) displays the Policy Details page for the `oracle/wss10_saml20_token_with_message_protection_service_policy`.

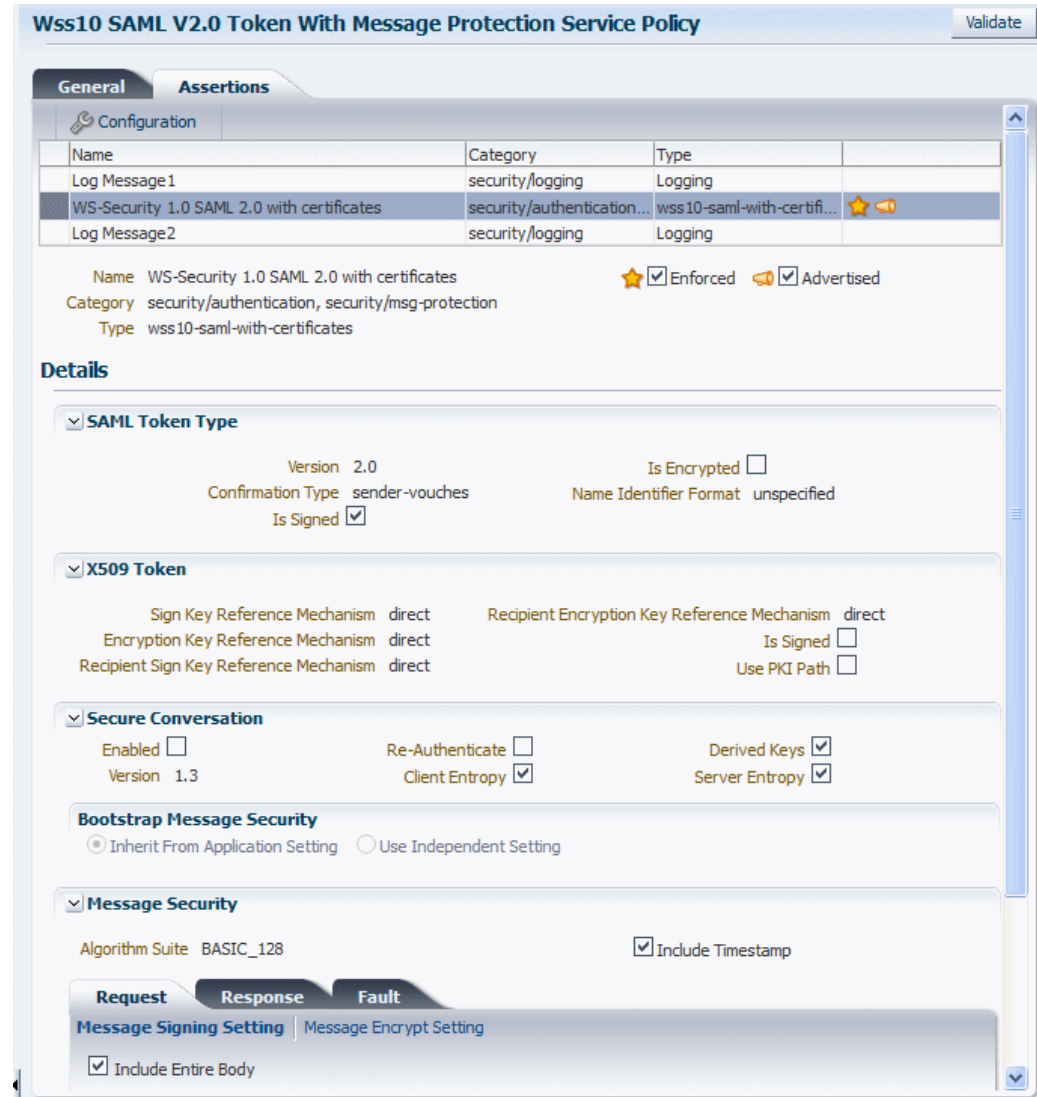
Figure 6-2 Policy Details Page with the General Tab Selected



The Policy Details page contains two tabs:

- The **General** tab (shown in [Figure 6-2](#)) displays information such as the policy name and display name, policy category, description, whether the policy is enabled, and the local optimization setting. The Attachment Attributes section provides details about the type of endpoints to which the policy can be attached, and the number of policy attachments. The Version Information section lists the version number of the policy, when it was last updated, and by whom. For user-created policies, you can also navigate to the Policy Version history page. For more information about policy versions, see "[Versioning Web Service Policies](#)".
- The **Assertions** tab includes a table that lists all of the assertions contained in the policy. Select the assertion name in the table to view the assertion details. The content displayed varies depending on the assertion selected. [Figure 6-3](#) displays the **Assertions** tab for the Wss10 SAML V2.0 Token With Message Protection Service Policy.

Figure 6-3 Policy Details Page with the Assertion Tab Selected



6.2.4 Creating and Editing Web Service Policies

Use the **WSM Policy Details** page, to create and edit web service policies.

Topics:

- [Creating a New Web Service Policy](#)
- [Cloning a Web Service Policy](#)
- [Creating Custom Policies](#)
- [Editing a Web Service Policy](#)

6.2.4.1 Creating a New Web Service Policy

Use the following procedure to create a new policy using one or more assertion templates.

Perform the following steps to create a new web service policy

1. Navigate to the WSM Policies page as described in "[Navigating to the WSM Policies Page](#)" and click **Create**. Alternatively, select **Actions** and then **Create**.

The Policy Details page includes two tabs: **General** and **Assertions**. The **General** tab is displayed by default.

2. On the **General** tab, optionally specify a unique name in the **Display Name** field to be used in the console to reference the policy. If you do not specify a display name, the policy name is used to reference the policy.
3. Enter a policy name in the **Name** field.

The policy name must include the directory in which the policy is located. For example, all predefined policies provided by Oracle are contained in the `oracle/` directory, such as `oracle/wss_http_token_service_policy`.

 **Note:**

Oracle recommends that you follow the policy naming conventions described in "Recommended Naming Conventions for Policies" in *Understanding Oracle Web Services Manager*.

You cannot edit the name of a policy once the policy is created. To change the policy name, you will need to clone the policy and assign it a different name.

4. Select the category to which the policy will belong from the **Category** drop-down menu.

 **Note:**

You can create policies in the Security and Management categories only.

5. Optionally, enter a brief description for the policy.
6. Select the **Enabled** option to enable the policy, if desired. Note that a policy that is not enabled is not enforced at run time.
7. Select the type of **Local Optimization** to be used for the policy from the drop-down menu. Available options are **off**, **on**, and **check-identity**. For more information about the local optimization feature, see "[Using Local Optimization with OWSM Policies \(SOA Composites\)](#)".
8. In the Attachment Attributes section of the page, specify the type of policy subjects to which the policy can be attached. From the **Applies To** menu, choose one of the following options:
 - **All**—Specifies that the policy can be attached to any type of policy subject, including service endpoints, client endpoints, and SOA components.
 - **SOA Components**—Specifies that the policy can be attached to SOA components.
 - **Service Bindings**—Specifies that the policy can be attached to web service and client endpoints. When you choose this option, in the **Service Category** field select whether the policy can be attached to web service endpoints, web service clients, or both.
9. Select the **Assertions** tab, and click **Add** to add assertions to your policy. For more information, see "[Adding Assertions to a Policy](#)".
10. Optionally, add an OR group to the policy. Select the **Add** menu then select **OR Group**. Then click **Add** to add the desired assertions to the OR group.

An OR group enables you to define multiple security subcategory options, only one of which can be executed. For example, a subset can contain both a SAML Token and a Username Token security/authentication subcategory assertion, so a web service application can use either one or the other, but not both.

For more information, see ["Adding an OR Group to a Policy"](#).

11. Configure the assertions as required by modifying the settings and configuration properties.
 - To edit the assertion settings, select the assertion and edit the settings in the Details section of the page.
 - To edit the configuration properties, click **Configuration**.

The list of configuration properties defined for the assertion are displayed.

Edit the configuration properties as described in ["Editing the Configuration Properties in an Assertion Template"](#) and click **OK**.

- To enable or advertise the assertion, select the **Enforced** or **Advertised** options, respectively.

For details about the settings and configuration properties for each assertion template, see [Oracle Web Services Manager Predefined Assertion Templates](#) .

12. When you have finished adding assertions to the policy, select the assertions in the table and use the **Move Up** and **Move Down** buttons to set the order in the policy. Assertions are invoked in the order in which they appear in the list.
13. Click **Save** to validate and save the policy.

If the policy is invalid, it is disabled as a precaution. After you correct the validation issues, you will have to enable the policy. For more information on policy validation, see ["Validating Web Service Policies"](#).

6.2.4.2 Cloning a Web Service Policy

You can create a new policy by cloning an existing web service policy. For example, you can create a copy of one of the read-only predefined policies and edit it to suit your needs. You can also create a copy of a policy that you have created. Once the policy is created, you can treat it like any other user-created policy, adding or deleting assertions, and modifying existing assertions.

Perform the following steps to clone a web service policy:

1. Navigate to the WSM Policies page as described in ["Navigating to the WSM Policies Page"](#).
2. Optionally, refine the list of policies displayed using Search, as described in ["Using Advanced Search"](#).
3. Select the policy to be cloned from the list of policies and click **Create Like**. Alternatively, select **Actions** and then **Create Like**.

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

 **Note:**

Oracle recommends that you follow the policy naming conventions described in "Recommended Naming Conventions for Policies" in *Understanding Oracle Web Services Manager*.

You cannot edit the name of a policy once the policy is created. To change the policy name, you will need to clone the policy and assign it a different name.

4. Modify the policy as required, including the assertions.
For details about adding assertions to the policy, see "[Adding Assertions to a Policy](#)".
For details about adding an OR group to the policy, see "[Adding an OR Group to a Policy](#)".
5. Click **Save** to validate and save the policy.
If the policy is invalid, it is disabled as a precaution. After you correct the validation issues, you will have to enable the policy. For more information on policy validation, see "[Validating Web Service Policies](#)".

6.2.4.3 Creating Custom Policies

You can create custom policies using custom assertions.

For more information and procedures about how to create both custom assertions and policies, see *Creating Custom Assertions in Developing Extensible Applications for Oracle Web Services Manager*.

6.2.4.4 Editing a Web Service Policy

You can edit a user-created policy as described in this topic.

 **Note:**

The predefined policies that are provided with OWSM are read-only and cannot be edited. To edit a predefined policy you can clone it and then edit the cloned version.

The changes that you make to the policy 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. For more information about policy versioning, see "[Versioning Web Service Policies](#)".

Perform the following steps to edit a web service policy:

1. Navigate to the WSM Policies page as described in "[Navigating to the WSM Policies Page](#)".
Optionally, refine the list of policies displayed using Search, as described in "[Using Advanced Search](#)".
2. Select the policy to be edited from the list of policies and click **Open**. Alternatively, select **Actions** and then **Open**.

The Policy Details page is displayed. For predefined policies, this page is read-only. However, for user-created policies, you can edit the policy from this page. For more information about the Policy Details page, see "[Viewing the Details of a Web Service Policy](#)".

3. Select the **General** tab and edit as follows:
 - Edit the display name and description, if desired. You cannot edit the policy name. To change the name of a policy, you will need to clone it and assign it a different name.
 - Edit the remaining fields on the tab as required, including enabling or disabling the policy, specifying local optimization, or modifying the type of policy subjects to which the policy can be attached.
4. Select the **Assertions** tab and edit as follows:
 - Modify the assertion settings and configuration properties as required. To modify the assertion settings, select the assertion in the table and edit the settings as required in the Details section of the page. To edit the configuration properties, click **Configuration** and edit the properties as required in the Configuration table. To enable or advertise the assertion, select the **Enforced** or **Advertised** options, respectively.
 - Add assertions or OR groups as required, as described in "[Adding Assertions to a Policy](#)" and "[Adding an OR Group to a Policy](#)", respectively.
 - Delete assertions or OR groups as required. To do so, select the assertion or OR group in the table and click **Delete**.

For details about the assertions in each predefined policy, see [Oracle Web Services Manager Predefined Policies](#).
5. Click **Validate** to validate the policy.
6. Click **Save** to save the changes.

6.2.5 Using Local Optimization with OWSM Policies (SOA Composites)

OWSM supports an Oracle SOA Suite 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. If a policy is attached to the web service binding, the policy may not be invoked if local optimization is used.

Topics:

- "[Viewing the Default Local Optimization Setting in OWSM Policies](#)"
- "[Controlling When Local Optimization is Used](#)"

For details about the SOA local optimization feature, see "Configuring Local Optimization" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

6.2.5.1 Viewing the Default Local Optimization Setting in OWSM Policies

By default, each of the OWSM predefined policies includes a local optimization property. The default setting for the Local Optimization property is displayed on the **General** tab of each policy. Procedures for viewing the details of a policy using Fusion Middleware Control, including the Local Optimization property setting, are described in "[Viewing the Details of a Web Service Policy](#)".

There are three possible settings for the Local Optimization property:

- `on` — Local optimization is turned on in the policy and the policy is *not* applied at runtime.

- `off` — Local optimization is turned off and the policy is applied at runtime. The request goes through the usual WS/SOAP/HTTP process.
- `check-identity` — Local optimization is used only if a JAAS subject exists in the current thread, indicating that authentication has already succeeded. If the JAAS subject does not exist in the thread, the request goes through the usual WS/SOAP/HTTP process.

6.2.5.2 Controlling When Local Optimization is Used

There are two ways to control the local optimization feature, and they have different scopes:

- At the composite level, by adding the `oracle.webservices.local.optimization` property in the binding section of the `composite.xml` file. The following values are supported:
 - `true` -- (Default value). Local optimization is used if the attached policy is configured to use it. For a description of the local optimization property settings, see "[Viewing the Default Local Optimization Setting in OWSM Policies](#)". If optimization is used, the policy is not applied.
 - `false` -- Local optimization is not used, regardless of the how the local optimization property is configured in the policy. 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`.

For more information, see "Policy Attachments and Local Optimization in Composite-to-Composite Invocations" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

- At the policy level, by configuring the local optimization property for a policy. The possible settings for the local optimization property are described in "[Viewing the Default Local Optimization Setting in OWSM Policies](#)". The policy-level property controls the optimization wherever the policy is used, unless it has been overridden by the composite-level property.

Note:

Predefined policies from Oracle are read-only and cannot be modified. If you clone a predefined policy, Oracle recommends that you do not change the local optimization setting. Doing so may prevent the policy from being invoked, resulting in unexpected behavior. If you create a new policy, however, you can set this property as required for your environment.

If a policy is attached to a web service, the policy may not be invoked if local optimization is used. Therefore, for each new policy that you create, you need to decide whether you want to use local optimization.

6.2.6 Generating Client Policies from a WSDL

After you have configured a web service, you can use the web service WSDL to generate compatible client policies with the parameters required to call that service. Note that only

assertions that are advertised in a policy can be used to generate equivalent client assertions in the client policy.

 **Note:**

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.

When you generate a client policy, it is populated with the client assertion that is the matching pair to the advertised service assertion. For example, if the service policy in the WSDL contained the *oracle/wss_http_token_service_template*, then the generated client policy is populated with its counterpart, *oracle/wss_http_token_client_template*.

Before editing the policy, you must first save it. After you have made the desired changes to the policy, you can access it from the WSM Policies 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.

Perform the following steps to create 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. In the Oracle WSDL URL field, enter the URL to the web service WSDL using the following format: *Web_service_endpoint?orawsdl*, where *Web_service_endpoint* is the URL to the web service, for example `http://my-host:port/jaxwsejb/Calculator?wsdl`.

 **Note:**

You must use *?orawsdl*, instead of *?wsdl*, to get the WSDL that is used to generate the corresponding client policy.

The service policy information in the Oracle WSDL published for the web service is used as the basis for generating the initial client policies.

3. If HTTP authentication is required, select **Authentication** and provide the username and password in the appropriate fields.
4. Click **Fetch**.
5. Select the service and port from the drop-down lists.

The service and port combination define the endpoint to which the policies can be attached in the *orawsdl*.

6. Click **Generate Policies**.

The client policies corresponding to the service policy specified in the *wsdl* for the service and port are listed in the Generated Policy Results table as shown in [Figure 6-4](#).

Figure 6-4 Generate Client Policies from WSDL

Oracle WSDL

Enter the URL of the Web Service Endpoint. you must enter ?orawSDL instead of ?wsdl, to the WSDL that is used to generate the corresponding client policy.

* Oracle WSDL URL

Format: http://my-host:port/path/to/may-orawSDL.wsdl?orawSDL

Authentication

Username

Password

Pick the service and port from the list and click the Generate button

Service Port

Generated Policy Results

Category	Policy Name	Display Name
Security	oracle/wss11_saml_token_identity_switcd	Wss11 Saml Token Identity Switch With Messag
Security	oracle/wss11_saml_token_with_message	Wss11 Saml Token With Message Protection Cli
Security	oracle/wss11_username_token_with_me:	Wss11 Username Token With Message Protecti
Security	oracle/wss_saml_token_bearer_over_ssl	Wss SAML Token(confirmation method as beare
Security	oracle/wss_username_token_over_ssl_cli	Wss Username Token Over SSL Client Policy

7. Optionally, select the policy and display name in the Generated Policy Results table and edit as desired.
8. Click **Add Policies**.
The Generate Client Policies page is displayed.
The generated policies are listed in the table and their status is indicated as Not Saved.
9. Click **Save All** to save all the policies, or select individual policies and click **Save**.

Note:

You must save the policies before you can edit them.

10. To edit a policy, select the policy in the table and click **Open**.
11. In the Policy Details page, edit the policy as necessary.
12. Click **Apply** to save the changes to your policy.

13. You are returned to the Generated Client Policies page. Edit and save the other policies as needed.

Once the policy is saved, you can navigate to the WSM Policies page and find the policy in the list of policies.

6.2.7 Adding Assertions to a Policy

You can add assertions to a user-created policy during policy creation or editing. You cannot add assertions to the predefined policies provided with OWSM. The predefined policies are read-only and cannot be modified.

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 Service Policies](#)".

Perform the following steps to add an assertion to a policy:

1. Navigate to the Policy Details page for the policy to which you want to add assertions.
2. Select the **Assertions** tab.
3. Click **Add** or select **Assertion** from the **Add** menu.

The Add Assertion page is displayed. The assertions available for that policy are displayed in the Search Results table, organized by Template Name. Optionally, use the **View** menu to display the Display Name column, or to change the order of the columns.

4. Select an assertion from the table, or provide search parameters in the **Name** and **Category** fields and click **Search**. The results that match the search criteria are displayed in the Search Results table. In the Search Results table, select the assertion or assertions to be added to the policy and click **Add Selected**. To add all the listed assertions to the policy, click **Add All**.

The selected assertions are displayed in the Selected Assertion Templates table. The assertions are displayed using the Template Name. Optionally, use the **View** menu to display the Template Display Name column, or to change the order of the columns.

5. In the Selected Assertion Templates table, optionally edit the names for the added assertions in the **Assertion Name** field.
6. Review the selections in the Selected Assertion Templates table. To remove one or more assertions from this table, click **Remove Selected** or **Remove All**. When you have confirmed the assertion selection, click **Add Assertion**.

The added assertions are listed in a table in the **Assertion** tab.

For details about the OWSM assertion templates, see [Oracle Web Services Manager Predefined Assertion Templates](#).

7. To configure the assertion, select the assertion and edit the settings as required in the Details section of the page.
8. To enable or advertise the assertion, select the **Enforced** or **Advertised** options, respectively.
9. To edit the configuration properties, click **Configuration**.

The list of configuration properties defined for the assertion are displayed.

10. Edit the Configuration properties and click **OK**.

For details about the configuration properties for each assertion template, see [Oracle Web Services Manager Predefined Assertion Templates](#).

Note that you can edit only the **Value**, and **Description** fields. The **Name**, **Type**, and **Default Value** property settings defined in the assertion template cannot be changed, and are displayed as read only. For details about these properties, see ["Editing the Configuration Properties in an Assertion Template"](#).

11. When you have finished adding assertions to the policy, select the assertions in the table and use **Move Up** and **Move Down** buttons to set the order in the policy. Assertions are invoked in the order in which they appear in the list.
12. When you are done, click **Save** to save the policy.

6.2.8 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)" in *Understanding Oracle Web Services Manager*.

You can add only one OR group to a policy. Once you have added an OR Group, the **OR Group** option is greyed out.

You add an OR group from the Policy Details page.

Perform the following steps to add an OR group to a policy:

1. Navigate to the Policy Details page for the policy to which you want to add the OR group.
2. Select the **Assertions** tab.
3. Select **OR Group** from the **Add** menu.

An **OR Group** row is added to the assertions table.

4. Select **Assertion to OR Group** from the **Add** menu. Notice that the **OR Group** is now greyed out on the menu, so you cannot add any additional OR groups.

Note:

If you click **Add** or select **Assertion** from the **Add** menu, the assertion will be added *outside* the OR group.

The Add Assertion search page is displayed.

5. Select one or more assertions from the Search Results table, or provide search parameters in the **Name** and **Category** fields and click **Search**. The results that match the search criteria are displayed in the Search Results table.

For details about the OWSM assertion templates, see [Oracle Web Services Manager Predefined Assertion Templates](#).

6. In the Search Results table, select the assertion or assertions to be added to the OR Group and click **Add Selected**. The selected assertions are displayed in the Selected Assertion Templates table.
7. In the Selected Assertion Templates table, optionally provide display names for the added assertions in the **Assertion Name** field.

- Review the selections in the Selected Assertion Template table. To remove one or more assertions from this table, click **Remove Selected** or **Remove All**. When you have confirmed the assertion selection, click **Add Assertion**.

The added assertions are listed under the OR Group in the list of assertions in the **Assertion** tab.

 **Note:**

The values for the WS-Policy attributes `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. For more information about WS-Policy attributes, see "[wsp:Policy Element](#)".

- To add additional assertions to the OR group, repeat steps 4 through 8.
 - Configure the assertions as required by modifying the settings and configuration properties.
 - To edit the assertion settings, select the assertion and edit the settings in the Details section of the page.
 - To edit the configuration properties, click **Configuration**.

The list of configuration properties defined for the assertion are displayed.

Edit the configuration properties as described in "[Editing the Configuration Properties in an Assertion Template](#)" and click **OK**.
 - To enable or advertise the assertion, select the **Enforced** or **Advertised** options, respectively.
- For details about the settings and configuration properties for each assertion template, see [Oracle Web Services Manager Predefined Assertion Templates](#) .
- When you have finished adding assertions to the OR group, select the assertions and use **Move Up** and **Move Down** to order them as needed. Assertions are considered for invocation in the order that they appear on the list.
 - 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**.
 - When you are done, click **Save** to save the policy.

6.2.9 Importing Web Service Policies

Follow the procedure in this section to import one or more user-created policies into the OWSM repository. Once the policies are imported, you can attach them to web services and make changes to them.

For more information on importing web service policies, see "[Understanding the Different Mechanisms for Importing and Exporting Policies](#)".

 **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.

Perform the following steps to import one or more web service policies:

1. Navigate to the WSM Policies page as described in "[Navigating to the WSM Policies Page](#)".
2. Click **Import**.

You are prompted to provide the name of a zip archive file containing the policies to be imported.

 **Note:**

The policies to be imported must use the following directory structure in the zip archive:

```
META-INF/policies/policyname
```

Within this directory structure, *policyname* includes the directory in which the policy is located.

In 11g, policies were exported as XML files. If you are importing a policy that you exported from an 11g domain, you must add the file to a zip archive using the directory structure specified above.

3. In the Import window, enter the path and file name for the zip archive file in the **File Upload** field, or click **Browse** to navigate to the directory where the policies archive file is located, then select the zip archive file to be imported.

4. Click **Import**.

If an error is encountered with one of the policies, the import process stops. For example, if there are five policies to be imported and an error is encountered in the third one, the first two will be imported but the remaining policies will not.

An information window is displayed listing the policies that were imported. Click **OK** to close the window.

The imported policies are added to the list of policies in the WSM Policies page.

6.2.10 Exporting Web Service Policies

You may 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.

You can export web service policies that you have created as described in "[Creating and Editing Web Service Policies](#)". Predefined policies cannot be exported because the same read-only version of the policy will exist in the target environment. Once the policy is exported, you can import it to another repository, attach it to web services, make changes to it, and so forth.

For more information about exporting web service policies, see "[Understanding the Different Mechanisms for Importing and Exporting Policies](#)".

Use the following procedure to export a policy from the OWSM repository:

1. Navigate to the WSM Policies page, as described in "[Navigating to the WSM Policies Page](#)".
2. Optionally, refine the list of policies displayed using Search, as described in "[Searching for Policies in the WSM Policies Page](#)".

3. Select the policy or policies to be exported from the list of policies and click **Export**.

The policies are added to a zip archive file named `policyexport.zip` by default.

4. Specify a file name for the archive file, if desired, then select a location in your local directory to which you want to save the zip file and click **Save**.

The directory structure for each policy is maintained in the archive file using the following structure:

```
META-INF/policies/policyname
```

Within this directory structure, *policyname* includes the directory in which the policy is located.

6.2.11 Versioning Web Service Policies

Whenever a change to a user-created policy is saved, a new version of the policy is automatically created and the version number is incremented. The Policy Manager maintains the history of these changes, enabling you to go back to an earlier version.

Note:

Version control does not apply to the Oracle predefined policies because they are read only and cannot be modified.

Policy versioning requires that you use a database-based OWSM Repository. If you are using a file-based repository, versioning information is not maintained or displayed.

The recreation of a document (either by importing an existing document or by creating a new document) with the same name that already exists in the repository will result in increment of the version number.

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.

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 Policy Details page.

The following sections describe versioning in more detail:

- [Viewing the Version History of a Web Service Policy](#)
- [Changing the Current Version of a Policy](#)
- [Deleting Versions of a Web Service Policy](#)
- [Exporting a Version of a Policy](#)

6.2.11.1 Viewing the Version History of a Web Service Policy

You can view the version history for a web service policy from the Policy Version history page, which you can access from the **Policy Details** page.

Perform the following steps to view the version history for a policy:

1. Navigate to the Policy Details page for the policy as described in "[Viewing the Details of a Web Service Policy](#)".
2. Select the **General** tab for the policy, if it is not already selected.
3. In the Version Information section of the page, click **Versioning History**.

The Policy Version history for the page is displayed, as shown in [Figure 6-5](#). The policy versions appear in order in the version history table at the top of the page. The currently active policy has the highest version number, and is the only policy that can be attached to a policy subject. However, you can make an earlier version of a policy the active version.

Figure 6-5 Policy Version History Page

Policy Version history

Name: hq/wss10_saml20_token_service_policy_nolog
Display Name: Wss10 SAML V2.0 Token Service Policy_nolog

Make Current Delete Export

Current	Version	Version Date	Updated By	Description
<input checked="" type="checkbox"/>	3	Mon Aug 27 22:19:21 UTC 2012	weblogic	This policy authenticates users using credentials provided in SAML
<input type="checkbox"/>	2	Mon Aug 27 22:16:37 UTC 2012	weblogic	This policy authenticates users using credentials provided in SAML
<input type="checkbox"/>	1	Mon Aug 27 22:15:59 UTC 2012	weblogic	This policy authenticates users using credentials provided in SAML

General **Assertions**

Display Name: Wss10 SAML V2.0 Token Service Policy_nolog
Name: hq/wss10_saml20_token_service_policy_nolog
Category: Security
Description: This policy authenticates users using credentials provided in SAML V2.0 tokens in the WS-Security SOAP header. The credentials in the SAML 2.0 token are authenticated against a SAML 2.0 login module. This policy can be applied to any SOAP-based endpoint. ...
Enabled:
Local Optimization: check-identity

Attachment Attributes

Applies To: Service Bindings
Service Category: Service Endpoint
Attachment Count: 0

Version Information

Version Number: 3
Last Updated: Mon Aug 27 22:19:21 UTC 2012
Updated By: weblogic

6.2.11.2 Changing the Current Version of a Policy

Use this procedure to change the current version of the policy:

1. In the Version Information section of the policy detail page, click **Versioning History** to display the Policy Version history page.
2. In the policy version table, select the version to be made current and click **Make Current**.

The selected policy version becomes the current active policy and the current version number is incremented by 1. The earlier version of the policy is retained.

6.2.11.3 Deleting Versions of a Web Service Policy

Use the following procedure to delete 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 a Web Service Policy](#)".

Perform the following steps to delete the version of a web service policy:

1. In the Version Information section of the policy detail page, click **Versioning History** to display the Policy Version history page.
2. In the policy version table, select the version or versions to be deleted and click **Delete**.
3. In the Confirm Policy Version Deletion box, click **OK**.

The selected policy version(s) is deleted from the OWSM Repository and the Policy History table.

6.2.11.4 Exporting a Version of a Policy

Use the following procedure to export a version of the policy:

1. In the Version Information section of the policy detail page, click **Versioning History** to display the Policy Version history page.
2. In the policy version table, select the version to be made exported and click **Export**.
You are prompted to open or save the file.
3. Select **Save File** and click **OK**.
4. Navigate to the local directory to which you want to save the file and update the filename as desired.
5. Click **Save**.

6.2.12 Deleting a Web Service Policy

Before you delete a policy, Oracle recommends that you verify that the policy is not attached to any policy subjects. 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.



Note:

Only user-created policies can be deleted. The predefined policies delivered with OWSM are read only and cannot be edited or deleted.

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 "[Deleting Versions of a Web Service Policy](#)".

Perform the following steps to delete a user-created web service policy:

1. Navigate to the WSM Policies page as described in "[Navigating to the WSM Policies Page](#)".
Optionally, refine the list of policies displayed using Search, as described in "[Using Advanced Search](#)".
2. From the WSM Policies page, select the policy to be deleted from the list of policies and click **Delete**. Alternatively, select **Actions** and then **Delete**.
3. A dialog box appears asking you to confirm the deletion. Click **Delete**.

6.3 Validating Web Service Policies

There are restrictions on the type and number of policy assertions that are permitted in a web service policy. 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`, as shown in Figure 6-6.

Figure 6-6 Security Assertion with Two Subcategories



- 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 create a new policy or edit a user-created policy, 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.

Perform the following steps to validate a policy:

1. On the Policy Details page of the policy being viewed or edited, click **Validate**.
If the validation is successful, the `Policy is Valid` message appears.
If the validation is not successful, the resulting error message describes the problem. Make the necessary corrections, then revalidate the policy.
2. Once the policy validates successfully, click **Save** to save the policy.

6.4 Managing Policy Assertion Templates

OWSM includes a set of predefined assertion templates that you can use to construct policies. The predefined assertion templates are read only and cannot be modified, but you can clone them to create new assertion templates, if needed, to satisfy a specific requirement.

For additional information, see "Building Policies Using Policy Assertions" in *Understanding Oracle Web Services Manager*.

If the functionality you require, such as support for a non-standard security token, is not provided out of the box, OWSM allows you to define custom policy assertions. For details, see *Creating Custom Assertions*.

You can add one or more assertions to a user-created policy as described in "[Adding Assertions to a Policy](#)". You cannot add assertions to the predefined policies that are provided with OWSM because they are read-only and cannot be modified. Assertions are executed in the order in which they are listed in the policy.

For details about the predefined assertion templates, see [Oracle Web Services Manager Predefined Assertion Templates](#).

The following sections provide more detail about managing policy assertion templates:

- [About Navigating to the Assertion Templates Page](#)
- [Understanding Search Options on the Assertion Templates Page](#)
- [Viewing the Details of an Assertion Template](#)
- [Cloning an Assertion Template](#)
- [Editing an Assertion Template](#)
- [Editing the Configuration Properties in an Assertion Template](#)
- [Exporting an Assertion Template](#)
- [Importing an Assertion Template](#)
- [Deleting an Assertion Template](#)

6.4.1 About Navigating to the Assertion Templates Page

You can manage your assertion templates at the domain level from the Assertion Templates page. From this page, you can copy, edit, and delete, import, and export assertion templates.

6.4.2 Understanding Search Options on the Assertion Templates Page

You can search for assertion templates in the Assertion Templates page using the advanced search feature, the Query by Example filter, or a combination of the two to refine the search.

Details are provided in the following sections:

- [Searching for an Assertion Template Using Advanced Search](#)
- [Searching for an Assertion Template Using the Query by Example Filter](#)

6.4.2.1 Searching for an Assertion Template Using Advanced Search

In the **Assertion Templates** page, you can reduce the number of assertion templates that are returned by specifying the appropriate search criteria. To do so:

1. Navigate to the Assertion Templates page as described in "[About Navigating to the Assertion Templates Page](#)".
2. In the Search pane, specify the criteria to use in the search:
 - In the **Assertion Name** field, enter an assertion template name or part of a name and select the operator to use to refine the search. Available operators are Starts with, Ends with, Equals, and Contains. For example, to search for message protection assertion templates only, select the **Contains** operator, and enter `message` in the **Assertion Name** field.

You can use percent `%` as a wildcard, any place in the name. Asterisk `*` is not recognized as a wildcard and is treated as plain text. Searches are case-insensitive.

- In the **Category** field, select the category to which the assertion template belongs. The options are: All, Management, Security, Reliable Messaging, MTOM Attachments, WS-Addressing, Make Connection, Atomic Transactions, Configuration, and SOAP Over JMS Transport.

Alternatively, you can select one of the previously saved searches from the **Saved Search** drop-down menu. The search parameters automatically populate the search fields. If the **Run automatically** option is specified for the saved search, it runs automatically and the results are displayed in the assertion templates table.

3. Optionally, refine the search using the Query By Example filter, as described in "[Searching for an Assertion Template Using the Query by Example Filter](#)". Note that when you combine the two search types, the data entered into the Query By Example fields is appended using the AND operator to the data specified in the Search fields.

4. Click **Search**.

The Assertion Templates table is refreshed to include only those assertion templates that match the specified search criteria. If you did not refine the search using Query by Example, all message protection assertion templates are shown in the list. If you refined the search for client assertion templates only using Query by Example, the list displayed includes only client message protection assertion templates.

5. Optionally, click **Save...** to save the search criteria in the repository. Note that only the values specified in the Advanced Search fields are saved; the values specified in the Query By Example fields are not included in the saved searches.

In the Create Saved Search window, enter a name for the search in the Name field. To use this saved search as the default selection for future searches, select **Set as default**. To execute the search automatically when it is selected, select **Run automatically**.

To modify previously saved searches, click **Personalize...** from the **Saved Search** drop-down menu. In the Personalize Saved Searches window, select the saved search from the drop down menu and edit as required. Click **Apply**. When you are finished editing all of the searches, click **OK**.

6.4.2.2 Searching for an Assertion Template Using the Query by Example Filter

The Query by Example filter allows you to query a specific field and filter the results displayed in the table quickly and easily.

1. If the search fields are not displayed at the top of the Assertion Templates table, click the Query by Example icon. A search field is displayed above the Display Name, Category, and Name columns.
2. Enter the search criteria in the field above the column in which you want to search. The value entered is interpreted as a "contains" expression. That is, the value is wrapped in `%value%`, and will fetch all results that contain the value specified for that column. For example, to search for client assertion templates only, enter `client` in the search field above the **Name** column.
3. If you are using the Query by Example filter separately (not in conjunction with the advanced search fields), press **Enter**.

The list of assertion templates displayed in the table is filtered to display only the results that match the search criteria. Using the example specified in step 2, only client assertion templates are displayed.

 **Note:**

The Query by Example search fields can be used in conjunction with the advanced search fields to further refine the search results, as described in "[Searching for an Assertion Template Using Advanced Search](#)". When used together, the data entered into the Query By Example fields is appended using the AND operator to the data specified in the Search fields. You must use the **Search** button to get the combined results.

You need to manually clear the Query by Example search fields when you have completed the search.

6.4.3 Viewing the Details of an Assertion Template

Use these procedure to view the details of an assertion template. Predefined assertion templates from Oracle are read-only and cannot be modified. These assertion templates are displayed in read-only mode. User-created assertion templates are not read-only, and can be edited.

Perform the following steps to view the assertion template details:

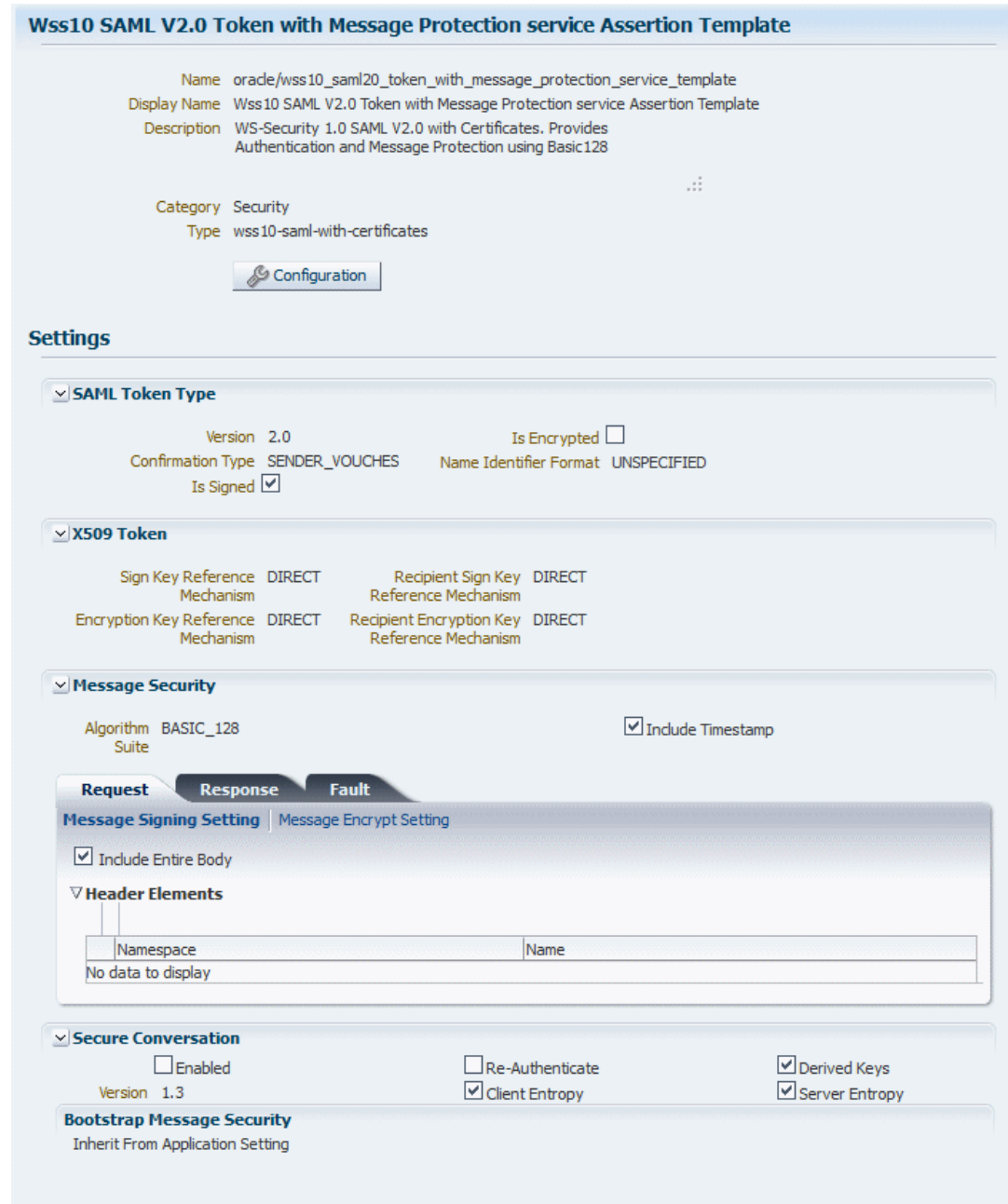
1. Navigate to the Assertion Templates page as described in "[About Navigating to the Assertion Templates Page](#)".

Optionally, refine the list of assertion templates displayed using Search, as described in "[Searching for an Assertion Template Using Advanced Search](#)".

2. Select the assertion template to be viewed from the list of assertion templates and click **Open**. Alternatively, select **Actions** and then **Open**.

[Figure 6-7](#) displays the Assertion Template Details page for the Wss10 SAML V2.0 Token with Message Protection service Assertion Template.

Figure 6-7 Assertion Template Details Page



3. Review the details of the assertion template.

General information about the assertion template is provided at the top of the page. Click **Configuration** to view the configuration properties for the template. The Settings section of the page displays the settings specific to that template. For details about the settings and configuration properties for each of the predefined assertion templates, [Oracle Web Services Manager Predefined Assertion Templates](#).

6.4.4 Naming Conventions for Assertion Templates

The same naming conventions used to name predefined policies are used to name the assertion templates.

The predefined 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`.

It is recommended that you follow the recommended naming conventions, and keep any assertion templates that you create in a directory that is separate from the `oracle` directory where the predefined assertion templates are located. You can organize your assertion templates at the root level, in a directory other than `oracle`, or in subdirectories.

For more information about the naming conventions for predefined policies, see "Recommended Naming Conventions for Policies" in *Understanding Oracle Web Services Manager*.

6.4.5 Cloning an Assertion Template

You can create a new assertion template using an existing template as the base. Select the assertion template that most closely matches the desired behavior, make a copy of it using the **Create Like** feature, then make any changes required to get the new behavior.

Perform the following steps to clone a web service policy:

1. Navigate to the Assertion Templates page as described in "[About Navigating to the Assertion Templates Page](#)".
2. Optionally, refine the list of assertion templates displayed using Search, as described in "[Understanding Search Options on the Assertion Templates Page](#)".
3. Select the assertion template to be cloned from the list of assertion templates and click **Create Like**. Alternatively, select **Actions** and then **Create Like**.

The Assertion Template Details page is displayed.

4. Edit the name and display name for the assertion template and, optionally, enter a brief description.

The word *Copy* is appended to the name and display name of the cloned assertion template and, by default, this is the name assigned to the new assertion template.

It is recommended that you change the name of this new assertion template to be more meaningful in your environment. For more information, see "[Naming Conventions for Assertion Templates](#)".

Note:

You cannot edit the name of an assertion template after it is created. To change the assertion template name, you will need to clone the assertion template and assign it a different name.

5. Modify the assertion template settings and configuration properties as required. For details about the settings and configuration properties in each of the predefined assertion templates, see [Oracle Web Services Manager Predefined Assertion Templates](#). For details about modifying the configuration properties, see "[Editing the Configuration Properties in an Assertion Template](#)".
6. Click **Save** to save the new assertion template.

6.4.6 Editing an Assertion Template

You can edit a user-created assertion template as described in the following procedure.

 **Note:**

The predefined assertion templates that are provided with OWSM are read-only and cannot be edited. To edit a predefined template you can clone it and then edit the cloned version.

Perform the following steps to edit an assertion template:

1. Navigate to the Assertion Templates page as described in "[About Navigating to the Assertion Templates Page](#)".
Optionally, refine the list of assertion templates displayed using Search, as described in "[Searching for an Assertion Template Using Advanced Search](#)".
2. Select the assertion template to be edited from the list of assertion templates and click **Open**. Alternatively, select **Actions** and then **Open**.
3. Edit the display name and description, if desired. You cannot edit the assertion template name. To change the name of an assertion template you will need to clone it and assign it a different name. and assertion template as required and click **Save**.
4. Edit the settings as required.
For details about the settings and configuration properties for each of the predefined assertion templates, see [Oracle Web Services Manager Predefined Assertion Templates](#) .
5. Click **Configuration** to edit the configuration properties.
To delete a property, select the property in the table and click **Delete**.
6. Click **OK** to accept the configuration property changes.
7. Click **Save** to save the assertion template.

6.4.7 Editing the Configuration Properties in an Assertion Template

If you have cloned one of the predefined assertion templates, you can modify the configuration properties 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.

 **Note:**

You cannot modify the configuration properties in the predefined assertion templates because they are read-only and cannot be modified.

When you clone an assertion template, or edit a cloned assertion template, you can configure the following settings for each property:

- **Description**—Description of the property.
- **Value**—Current value.
- **Default**—Default value. This value is used if the **Value** field is not set.
- **Type**—Can be one of the following:
 - **Constant**—Property cannot be overridden.
 - **Required**—Property is required and can be overridden. This value determines if the **Value** property needs to have a value or it could be left blank.
 - **Optional**—Property is optional and can be overridden.

Perform the following steps to configure the properties:

1. In the assertion template being cloned or edited, click **Configuration**.
The Configuration window displays the list of properties for the template.
2. Select the property from the list and modify the fields as required. Note that the Name of an existing property cannot be changed.
3. Add or delete configuration properties as required.
To add a configuration property, click **Add**. In the blank row that appears, provide a name for the property. The remaining fields are optional. However, if you select Type **required**, then you must provide a value for the property.
To delete a configuration property, select the property in the table and click **Delete**.
4. When you have finished changing the configuration properties, click **OK**.
5. Click **Save** to save the changes in the assertion template.

 **Note:**

When you add an assertion to a policy, as described in "[Adding Assertions to a Policy](#)", you can modify the **Value**, **Default**, and **Description** configuration properties to match your environment. The **Name** and **Type** configuration properties defined in the assertion template cannot be changed, and are not editable fields in the table.

6.4.8 Exporting an Assertion Template

You can export one or more assertion templates that you have created. Predefined assertion templates can not be exported because the same read-only version of the template will exist in the target environment. After you have exported the assertion templates, you can then copy them to a new directory if desired, or import them into another repository.

Exporting one or more assertion templates is described in "[Cloning an Assertion Template](#)".

Perform the following steps to export one or more assertion templates:

1. Navigate to the Assertions Templates page, as described in "[About Navigating to the Assertion Templates Page](#)".
2. Optionally, refine the list of assertion templates displayed using Search, as described in "[Understanding Search Options on the Assertion Templates Page](#)".
3. Select the assertion template or templates to be exported from the list of assertion templates and click **Export**.

The assertion templates are added to a zip archive file named `assertiontemplatesexport.zip` by default.

4. Specify a file name for the archive file, if desired, then select a location in your local directory to which you want to save the zip file and click **Save**.

The directory structure for each assertion template is maintained in the archive file using the following structure:

```
META-INF/assertiontemplates/assertiontemplatename
```

Within this directory structure, *assertiontemplatename* includes the directory in which the template is located and represent the values you specified when you created the template.

6.4.9 Importing an Assertion Template

Follow the steps in this section to import a zip archive containing one or more user-created assertion templates. You can use this feature in combination with **Export** to move one or more assertion templates between different repositories. Once the assertion template is imported, you can add it to web service policies and make changes to it.

Perform the following steps to import an assertion template:

1. Navigate to the Assertions Templates page, as described in "[About Navigating to the Assertion Templates Page](#)".
2. Click **Import**.

You are prompted to provide the name of a zip file containing the assertion templates to be imported.

Note:

The assertion templates to be imported must use the following directory structure in the zip archive:

```
META-INF/assertiontemplates/assertiontemplatename
```

Within this directory structure, *assertiontemplatename* includes the directory in which the template is located.

In 11g, assertion templates were exported as XML files. If you are importing an assertion template that you exported from an 11g domain, you must add the file to a zip archive using the directory structure specified above.

3. In the Import window, enter the path and file name for the zip file in the **File Upload** field, or click **Browse** to navigate to the directory where the assertion template zip file is located, then select the zip file to be imported.
4. Click **Import**.

If an error is encountered with one of the assertion templates, the import process stops. For example, if there are five assertion templates to be imported and an error is encountered in the third one, the first two will be imported but the remaining assertion templates will not.

An information window is displayed listing the assertion templates that were imported. Click **OK** to close the window.

The imported assertion templates are added to the list of assertion templates on the Assertion Templates page.

6.4.10 Deleting an Assertion Template

Follow the steps in this section to delete an assertion template that you created or imported. The predefined assertion templates delivered with OWSM are read-only and cannot be deleted.

1. Navigate to the Assertions Templates page, as described in "[About Navigating to the Assertion Templates Page](#)".
2. Optionally, refine the list of assertion templates displayed using Search, as described in "[Understanding Search Options on the Assertion Templates Page](#)".
3. Select the assertion template to be deleted from the list of assertion templates and click **Delete**.

You are prompted to confirm that you want to delete the assertion template.

4. Confirm your selection and click **Delete**.

The selected assertion template is deleted from the list of assertion templates on the Assertion Templates page.

6.5 Managing Policies and Assertions

You can enable or disable policies, or assertions within a policy.

The following sections describe the different methods for enabling or disabling policies, or assertions within a policy:

- [Enabling or Disabling a Policy for all Policy Subjects](#)
- [Enabling or Disabling Assertions Within a Policy](#)

6.5.1 Enabling or Disabling a Policy for all Policy Subjects

When you create a policy, it is enabled by default unless it has validation errors. A user-created policy can be globally enabled or disabled from the Policy Details 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.

 **Note:**

You cannot disable a predefined policy from Oracle for all policy subjects. These policies are read-only and cannot be modified. You can, however, disable policy references to an individual subject. For more information, see "[Enabling or Disabling Directly Attached Policies Using Fusion Middleware Control](#)".

When you disable a policy from the Policy Details 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.

You may also selectively enable or disable a policy for a specific policy subject rather than for all policy subjects. For more information, see "[Enabling or Disabling Directly Attached Policies Using Fusion Middleware Control](#)".

Perform the following steps to enable or disable a user-created web service policy for all policy subjects:

1. Navigate to the WSM Policies page as described in "[Navigating to the WSM Policies Page](#)".

Optionally, refine the list of policies displayed using Search, as described in "[Using Advanced Search](#)".

2. Select the policy to be edited from the list of policies and click **Open**. Alternatively, select **Actions** and then **Open**.

The Policy Details page is displayed. For predefined policies, this page is read-only. However, for user-created policies, you can edit the policy from this page. For more information about the Policy Details page, see "[Viewing the Details of a Web Service Policy](#)".

3. Select the General tab if it is not already selected.
4. Select or deselect the **Enabled** box to enable or disable the policy, respectively.
5. Click **Save**.

6.5.2 Enabling or Disabling Assertions Within a Policy

You can 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.

Note:

You cannot disable an assertion in a predefined policy from Oracle. These policies are read-only and cannot be modified. To disable an assertion in a predefined policy, you need to clone it and then edit the cloned version.

For example, if you created a policy based on one of the read-only predefined web service security policies, it contains an instance of the Security Log 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" in *Administering Web Services*.)

Perform the following steps to enable or disable one or more assertions within a policy:

1. Navigate to the WSM Policies page as described in "[Navigating to the WSM Policies Page](#)".

Optionally, refine the list of policies displayed using Search, as described in "[Using Advanced Search](#)".

2. Select the policy to be edited from the list of policies and click **Open**. Alternatively, select **Actions** and then **Open**.

The Policy Details page is displayed. For predefined policies, this page is read-only. However, for user-created policies, you can edit the policy from this page. For more information about the policy details page, see "[Viewing the Details of a Web Service Policy](#)".

3. Select the **Assertions** tab.

4. Select the assertion in the table and select or deselect the **Enforced** box to enable or disable the assertion within the policy, respectively.
5. Click **Save**.

6.6 Analyzing Policy Usage

The policy usage feature described in this section requires that you use a database-based OWSM 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.

Perform the following steps to perform a usage analysis

1. Navigate to the WSM Policies page as described in "[Navigating to the WSM Policies Page](#)".

Optionally, refine the list of policies displayed using Search, as described in "[Using Advanced Search](#)".

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 column for the selected policy to display the Usage Analysis page.

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 policy subjects include OWSM Repository Documents and the subject types listed in "Understanding Policy Subjects" in *Understanding Oracle Web Services Manager*. 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.

4. In cases where multiple domains share the same OWSM Repository to store OWSM 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.

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 Directly Attached Policies Using Fusion Middleware Control"](#) and ["Enabling or Disabling a Policy for all Policy Subjects"](#).
- You must invoke an ADF DC client to display an accurate policy usage count.

6.7 About Advertising Policy Assertions

You can enable the advertisement of a policy assertion within the WSDL file.

The advertisement of a policy assertion within the WSDL file is enabled by selecting the **Advertised** option on the **Assertions** tab, as shown in [Figure 6-8](#), when performing any of the following tasks:

- Creating or editing an assertion template, as described in ["Creating and Editing Web Service Policies"](#).
- Adding an assertion or OR group to a policy, as described in:
 - ["Creating and Editing Web Service Policies"](#)
 - ["Adding Assertions to a Policy"](#)
 - ["Adding an OR Group to a Policy"](#)



Note:

Advertisement of policy assertions in a WADL file is not supported. The **Advertised** option has no effect when the associated policy is attached to a RESTful web service.

Figure 6-8 Enabling Advertising for Policy Assertions

s15_domain > WSM Policies > Policy Create Like

Http SAML Bearer V2.0 Token Client Policy_Copy Save Cancel

General **Assertions**

+ Add - Delete Move Up Move Down Configuration

Name	Category	Type	Options
Log Message1	security/logging	Logging	
Http SAML 2.0 Bearer Security	security/authentication	http-saml20-bearer-se...	★ ⚙
Log Message2	security/logging	Logging	

Name Http SAML 2.0 Bearer Security
Category security/authentication
Type http-saml20-bearer-security

★ Enforced **⚙ Advertised**

Details

Authentication Header

Mechanism saml20-bearer Header Name

6.8 About Advertising WS-Policy and WS-SecurityPolicy Versions

For a standard WSDL (`?wsdl`) file, 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 6-1 lists the valid version combinations.

Table 6-1 Policy Advertisement

Version Combination	Description
<code>?wsdl</code>	WS-Policy 1.2 and WS-SecurityPolicy 1.1
<code>?wsdl&wsp=1.5</code>	WS-Policy version 1.5 and WS-SecurityPolicy 1.3
<code>?wsdl&wssp=1.2</code>	WS-Policy versions 1.5 and WS-SecurityPolicy 1.2
<code>?wsdl&wssp=1.3</code>	WS-Policy versions 1.5 and WS-SecurityPolicy 1.3
<code>?wsdl&wsp=1.5&wssp=1.2</code>	WS-Policy 1.5 and WS-SecurityPolicy 1.2
<code>?wsdl&wsp=1.5&wssp=1.3</code>	WS-Policy 1.5 and WS-SecurityPolicy 1.3
<code>?wsdl&wsp=1.2&wssp=1.2</code>	WS-Policy 1.2 and WS-SecurityPolicy 1.2

Part III

Securing Web Services

This part describes how to configure your web service for message protection, SSL, authentication, and authorization. You can also integrate various hardware modules with Oracle Web Services Manager (OWSM).

Topics:

- [Configuring Message Protection for Web Services](#) , describes how to work with keystores and credential stores. It also describes how to use identity extensions and how to cache the nonce using Oracle Coherence.
- [Protecting Personally Identifiable Information](#), describes how to protect Personally Identifiable Information (PII) using the PII security policy.
- [Configuring Transport-Level Security \(SSL\)](#) , describes how to work with transport-level security in the form of SSL. It describes how to configure one-way and two-way SSL for WebLogic Server and for web service clients. It also describes how to configure SSL for Oracle HTTP Server.
- [Configuring Authorization Using Oracle Web Services Manager](#), describes how to configure authorization in OWSM, such as determining which resources to protect, and setting authorization permissions and an OPSS resource name. It also describes how to configure fine-grained authorization using OES.
- [Configuring Authentication Using Oracle Web Services Manager](#), describes how to configure authentication in OWSM, including authentication providers, digest authentication, SAML, Kerberos, ActiveDirectory, WS-Trust, and identity extensions.
- [Configuring Secure Conversation Using Oracle Web Services Manager](#) , describes how to configure WS-SecureConversation.
- [Integrating Hardware with Oracle Web Services Manager](#) , describes how to integrate Web Services Manager with various hardware security modules and for Oracle SPARC T4 and SPARC T5 cryptographic acceleration.

7

Configuring Message Protection for Web Services

This topic introduces and describes message protection configuration. You must configure your Fusion Middleware Control and WebLogic Server environments to use security policies. Message protection security policies or message protection and authentication with SSL security policies require you to set up keystores and truststores. (Authentication-only security policies do not require keys.)

Topics:

- [Overview of Message Protection Configuration for Web Services](#)
- [Overview of Configuring Keystores for Message Protection](#)
- [About Creating an Application-level Credential Map](#)
- [Understanding Service Identity Certificate Extensions](#)
- [Caching the Nonce with Oracle Coherence](#)
- [About Configuring Partial Encryption with Fusion Middleware Control](#)

7.1 Overview of Message Protection Configuration for Web Services

Message protection involves encrypting the message for message confidentiality and signing the message for message integrity. OWSM 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 summarize 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.
Note: Message protection-only policies do not authenticate or authorize the requester.
- Sign the message if you want message integrity.
- Encrypt the message if you want message confidentiality.
- 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 [Overview of Configuring Keystores for Message Protection](#).

To sign and encrypt SOAP messages, you use public and private signature and encryption keys that you store in the OWSM keystore for the WebLogic domain. The keystore configuration is domain wide: all web services and web service clients in the domain use this keystore.

For summaries of the message protection policies available in the current release, see ["Message Protection Only Policies"](#) and ["Message Protection and Authentication Policies"](#) in [Determining Which Predefined Policies to Use for a Web Service](#).

 **Note:**

The OWSM run time does not use the WebLogic Server keystore that is configured using the WebLogic Server Remote Console and used for SSL as documented in [About Configuring Keystores for SSL](#).

7.2 Overview of Configuring Keystores for Message Protection

Keys and the keystore provide the basis for configuring message protection. 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 the 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.

OWSM provides support for the following keystores:

- Keystore Service (KSS). For more information, see [Understanding OPSS Keystore Service for Message Protection](#).
- Java Keystore (JKS). For more information, see [Understanding Java Keystore for Message Protection](#).
- Hardware Security Module (HSM). For more information about using HSM as your keystore, see [Using Hardware Security Modules With OWSM](#).
- PKCS11. For more information about using the PKCS11 keystore, see [About Configuring OWSM for Oracle SPARC T5 and SPARC T4 Cryptographic Acceleration](#).

This section describes how to create JKS and KSS keystores, and how to populate these keystores with keys and certificates.

After these keystores are created, you need to configure the OWSM Keystore at the domain level. For more information, see the following topics:

- [OWSM Keystore Configuration Using Fusion Middleware Control](#).
- [Configuring the OWSM Keystore Using WLST](#).

This section contains the following topics:

- [Understanding OPSS Keystore Service for Message Protection](#)
- [Understanding Java Keystore for Message Protection](#)
- [Adding Keys and User Credentials to Configure the Credential Store](#)

7.2.1 Understanding OPSS Keystore Service for Message Protection

The OPSS Keystore Service provides a mechanism to manage keys and certificates for message security. This is the default approach for managing keys and certificates for message security.

For more information, see in "Managing Keys and Certificates with the Keystore Service" in *Securing Applications with Oracle Platform Security Services*.

This section contains the following topics:

- [Configuring Message Protection Using the OPSS Keystore Service](#)
- [Migrating a JKS Keystore Into the KSS Keystore](#)
- [Importing Certificates Into the KSS Keystore](#)
- [Overriding `keystore.sig.csf.key` and `keystore.enc.csf.key` Attributes](#)
- [Renewing or Regenerating the Expiring Certificates or Keys](#)

7.2.1.1 Configuring Message Protection Using the OPSS Keystore Service

This topic describes how to use the OPSS Keystore Service for message protection.

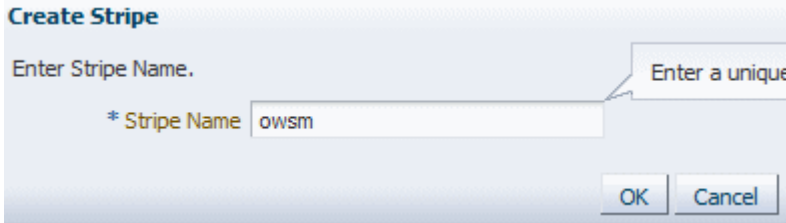
 **Note:**

In previous releases of OWSM, the JKS keystore was used by default. As of version 12.1.2, the OPSS Keystore Service is used by default for original installations. If you are upgrading from a prior release, your existing JKS keystore is used. 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.

To use the OPSS Keystore Service for message protection, perform the following steps:

1. Create a stripe and name it `owsm`.
 - a. In the content pane, select **WebLogic Domain**, then **Security**, and then **Keystore**.
 - b. Click **Create Stripe**. The Create Stripe screen is shown in [Figure 7-1](#).
 - c. Enter `owsm` and click **OK**.

Figure 7-1 Create Stripe



2. Create a keystore named `keystore` in the `owsm` stripe. (For more information, see "Creating a Keystore with Fusion Middleware Control" in *Securing Applications with Oracle Platform Security Services*.)
 - a. Select the `owsm` stripe you created and click **Create Keystore**.
The Create Keystore page is shown in [Figure 7-2](#).

Figure 7-2 Create Keystore

- b. Name this keystore `keystore`.
 - c. Set the protection type to **Policy**. (Password protected KSS keystores are not supported in this release.)
 - d. Clear the **Grant Permission** check box.
 - e. Do not specify a code base URL.
 - f. Click **OK**.
3. Select the keystore you just created and click **Manage**.
The Manage Certificates screen is shown in [Figure 7-3](#).

Figure 7-3 Manage Certificates

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 7-4](#).

Figure 7-4 Generate Keypair

- a. Specify an alias such as `orakey` for the key pair.
 - b. Specify other site-specific information as appropriate.
 - c. 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. Configure OWSM to use this keystore and alias as described in [Configuring OWSM to Use the KSS Keystore](#).
 6. Optionally, obtain trusted certificates, as described in "Importing a Certificate or Trusted Certificate with Fusion Middleware Control" in *Securing Applications with Oracle Platform Security Services*.

7.2.1.2 Migrating a JKS Keystore Into the KSS Keystore

If you have an existing JKS keystore, you can migrate one or more aliases from the JKS keystore to the KSS keystore. To do this, perform the following steps:

1. Make sure the aliases you want to import are in the JKS keystore. You can use `keytool` (or your tool of choice) to do this.

```
C:\keytool -list -keystore default-keystore.jks
Enter keystore password:
```

```
Keystore type: JKS
Keystore provider: SUN
```

```
Your keystore contains 1 entry
```

```
orakey, May 16, 2013, PrivateKeyEntry,
Certificate fingerprint (SHA1): DE:0C:37:D5:34:92:00:2E:30:D7:10:EF:93:A5:C0:04:
52:02:26:B7
```


2. Use WLST to import one or more aliases from the keystore using the `importKeyStore` script at the command line, as described in "Importing a Keystore at the Command Line."

For example:

```
connect("<wls adminuser>", "<wls admin password>", "t3://<host>:<port>")
svc = getOpssService(name='KeyStoreService')
:
:
svc.importKeyStore(appStripe='owsm', name='keystore', password='password',
aliases='orakey', keypasswords='password', type='JKS', permission=true,
filepath='<JKS keystore location>');
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean as
the root.
For more help, use help('domainRuntime')
Keystore imported. Check the logs if any entry was skipped.
:
:
wls:/rc6_domain/serverConfig> svc.listKeyStoreAliases(appStripe="owsm", name="key
store", password='', type="*")
Already in Domain Runtime Tree

democa
orakey
wls:/rc6_domain/serverConfig>
```

To import multiple keys using this command, specify a comma-separated list of aliases and key passwords. The KSS keystore does not use a password because it is protected using permissions.

3. Change the OWSM Domain Configuration Message Security screen to reflect the aliases from the KSS keystore that you want to use for the sign alias and encrypt alias, if applicable.

For example, if you imported the `orakey` and `oratest` aliases from your JKS keystore, you might select `orakey` as the sign alias and `oratest` as the encrypt alias.

See [Configuring OWSM to Use the KSS Keystore](#) for the steps to follow.

7.2.1.3 Importing Certificates Into the KSS Keystore

As described in "Managing Keys and Certificates with the Keystore Service" in *Securing Applications with Oracle Platform Security Services*, keystores can be exported and imported. KSS supports migration for JKS and JCEKS certificate formats.

In previous releases of OWSM, the JKS keystore was used by default. As of version 12.1.2, the KSS Keystore Service is used by default for original installations. If you are upgrading from a prior release, your existing JKS keystore is used.

If you want to import your existing JKS keystore certificates into the KSS keystore, see "Importing a Certificate or Trusted Certificate with Fusion Middleware Control" in *Securing Applications with Oracle Platform Security Services*.

7.2.1.4 Overriding `keystore.sig.csf.key` and `keystore.enc.csf.key` Attributes

If you override `keystore.sig.csf.key` and `keystore.enc.csf.key` as described in [Overview of Policy Configuration Overrides](#), you must change the attribute values to point directly to your chosen keystore alias instead of to `csf-key`.

The KSS Keystore Service does not use passwords and does not require you to configure the credential store. The alias for the sign and encrypt keys is used to store and retrieve the keys.

If you do not override `keystore.sig.csf.key` and `keystore.enc.csf.key`, no action on your part is required.

7.2.1.5 Renewing or Regenerating the Expiring Certificates or Keys

After the OPSS keystore CA is updated or a new CA is to be used, all those keys such as sign/encrypt key issued with previous CA must be regenerated. To renew the expiring certificates:

- Use the WLST command `listExpiringCertificates`.

Syntax:

```
svc.listExpiringCertificates(days='days', autorenew=true|false)
```

Where:

- `svc`: The service command object obtained through a call to `getOpssService()`.
- `days`: Only list certificates within these many days from expiration. You need to have a large number of days to have all the certificates renewed.
- `autorenew`: `true` for automatically renewing expiring certificates, `false` for only listing them.

Example:

```
svc.listExpiringCertificates(days='9999', autorenew=true)
```

You can regenerate the key pairs using Fusion Middleware Control and WLST. For more information, see "Managing Keys and Certificates" in *Securing Applications with Oracle Platform Security Services*.

7.2.2 Understanding Java Keystore for Message Protection

You have the option to use the Java keystore instead of the default KSS. The Java keystore contains the entities private keys and certificates associated with those private keys.

There is a single OWSM keystore per domain, and it is shared by all web services and clients running in the domain. Therefore, if you choose to configure the JKS keystore as described in this section, OWSM uses only that JKS keystore and ignores any KSS keystores already defined.

This section contains the following topics:

- [Generating Private Keys and Creating the Java Keystore](#)
- [Obtaining a Trusted Certificate and Importing it into the Keystore](#)
- [Configuring the OWSM Keystore](#)

7.2.2.1 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://docs.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 `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](#).
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
```

```
*****
*****
```

7.2.2.2 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, perform the following steps:

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](#), 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
```

7.2.2.3 Configuring the OWSM Keystore

After you have created the keystore, you need to configure OWSM at the domain level to use the JKS keystore. You can do so using Fusion Middleware Control or WLST commands. Refer to the following sections for more information:

Refer to the following sections for more information:

- [Configuring OWSM to Use the JKS Keystore](#)
- [Configuring the OWSM Keystore Using WLST](#)

7.2.3 Adding Keys and User Credentials to Configure the Credential Store

OWSM 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

OWSM uses the credential store to look up the following:

- Alias names and passwords for keys in the Java keystore.

For details about how OWSM uses the credential store to look up alias names and passwords from the Java keystore, see [Understanding Web Service Security Concepts](#).

- 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 [Overview of Policy Configuration Overrides](#). 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.

 **Note:**

If you configured the KSS keystore as described in [Configuring the OWSM Keystore](#), the credential store `oracle.wsm.security` map is not created for you. You must create it before you can use the credential store to store user credentials.

The credential store is not used for storing any keystore aliases when the KSS keystore is used.

If you configured the JKS keystore as described in [Configuring the OWSM Keystore](#), the aliases and passwords that you specified were 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.

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 "Setting up Private Keys and Certificates for Message Protection Policies" in *Understanding Oracle Web Services Manager*.

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 Java 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](#).

This section contains the following topics:

- [Adding Keys and User Credentials to the Credential Store Using Fusion Middleware Control](#)
- [Adding Keys and User Credentials to the Credential Store Using WLST](#)

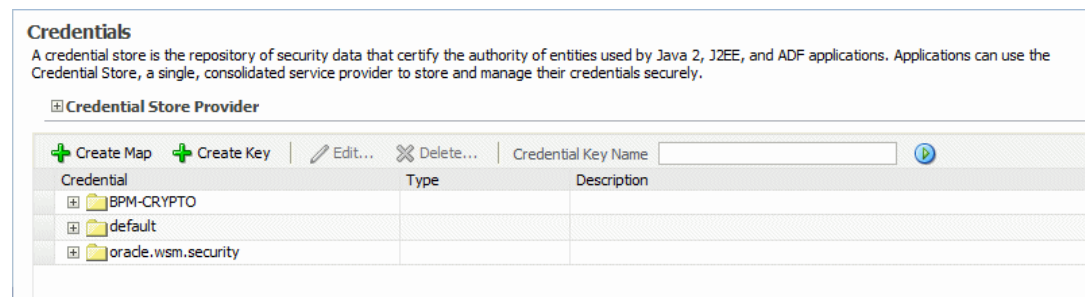
7.2.3.1 Adding Keys and User Credentials to the Credential Store Using Fusion Middleware Control

Follow these steps in Fusion Middleware Control to add keys and certificates to the credential store:

1. From **WebLogic Domain**, select **Security** then **Credentials**.

The Credentials page is displayed, as shown in [Figure 7-5](#).

Figure 7-5 Credential Store Provider Configuration Page

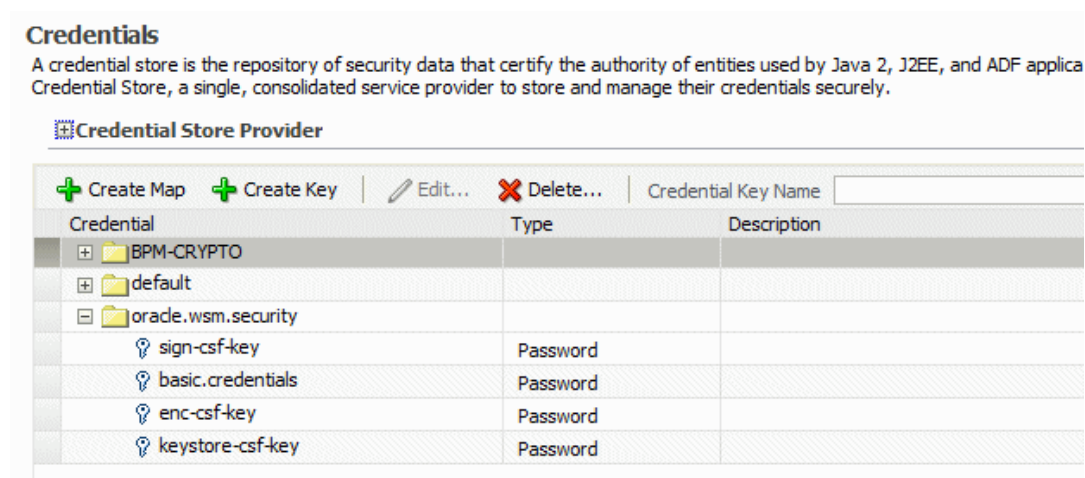


Note that in this configuration, the `oracle.wsm.security` credential map already exists in the credential store. This credential map was created when you configured the OWSM keystore as described in [Configuring the OWSM Keystore](#).

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.

2. Optionally, expand the `oracle.wsm.security` map in the Credential table to view the keys that have been configured in the map. [Figure 7-6](#) illustrates a sample OWSM credential store configuration.

Figure 7-6 Keys Configured in OWSM Credential Map



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 OWSM Java keystore.

3. 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 7-7](#).

Figure 7-7 Create Key Dialog Box

- 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.
4. 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 `AppPWord%`.
 - f. In the **Description** field, enter a description of for the entry, for example, `Username and Password for basic.credential key`.

- g. Click **OK**.
5. Restart the server.

7.2.3.2 Adding Keys and User Credentials to the Credential Store Using WLST

Follow these steps to add additional keys and user credentials to the credential store using WLST commands.

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services with Oracle Fusion Middleware*.
2. 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")
```

3. 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")
```

4. Repeat step 3 to create an entry for any additional aliases, for example `csfServiceB`, for the `ServiceB` alias.

5. 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="AppPWord%", desc="Key for ServiceA")
```

7.3 About 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.

The following topics describe creating application level credential map in detail:

- ["How CSF Keys Are Retrieved from an Application-level Credential Map"](#)
- ["About Permissions to Access an Application-level Credential Map"](#)
- ["Policies that Can Be Used to Access an Application-level CSF Map"](#)

For more information about configuring overrides, see [Overriding Policy Configuration Properties](#).

7.3.1 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.

More details about configuring the Oracle WSM keystore is described in [Configuring the OWSM Keystore](#).

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.

7.3.2 About Permissions to Access an Application-level Credential Map

To access an application-level credential map you must do the following settings.

Following sections describe the procedure to access an application-level credential map:

- [Configuring the csf.map Property Override](#)
- [About Granting CredentialAccessPermission to wsm-agent-core.jar](#)
- [About Grant WSIdentityPermission to wsm-agent-core.jar](#)

Also, refer to the "[Example of Granting Permission for Application-level In system-jazn-data.xml](#)" for more information.

7.3.2.1 Configuring the csf.map Property Override

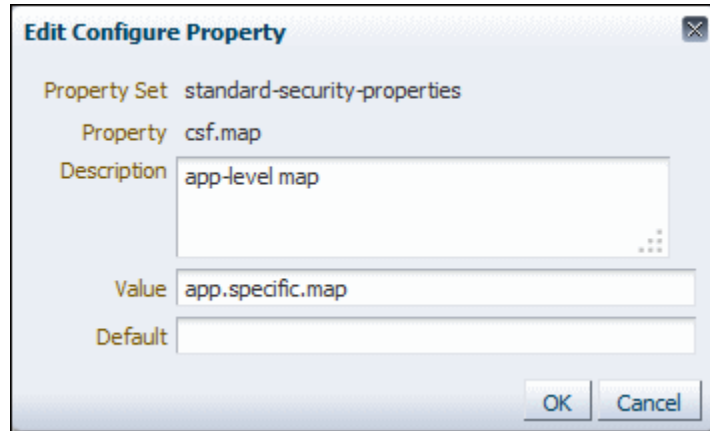
For an application to access its own credential map, the `csf.map` configuration override must be set for the policy that is attached to the application.

Perform the following steps to 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 WSM Policy Set Summary Page Using Fusion Middleware Control](#)
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 7-8](#) appears.

Figure 7-8 Edit Configure Property Dialog Box



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**.

7.3.2.2 About Granting 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.

Granting Credential Access permission can be done in one of the following ways:

- [Granting CredentialAccessPermission Using Oracle Enterprise Manager](#)
- [Granting CredentialAccessPermission Using WLST](#)

7.3.2.2.1 Granting CredentialAccessPermission 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 *Securing Applications with Oracle Platform Security Services*.

7.3.2.2 Granting CredentialAccessPermission Using WLST

You can grant `CredentialAccessPermission` to an application-level credential map using the `grantPermission` WLST command.

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 [Granting CredentialAccessPermission 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,permClass='oracle.security.jps.service.credstore.CredentialAccessPermission',permTarget='context=SYSTEM,mapName=application.csf.map.name,keyName=*',permActions='*')
```

For more information about this WLST command, see "Infrastructure Security Custom WLST Commands" in the *WLST Command Reference for WebLogic Server*.

7.3.2.3 About 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. This can be done in one of the following ways:

- [Granting WSIdentityPermission Using Oracle Enterprise Manager](#)
- [Granting WSIdentityPermission Using WLST](#)

7.3.2.3.1 Granting WSIdentityPermission 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 [About Granting 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 *Securing Applications with Oracle Platform Security Services*.

7.3.2.3.2 Granting WSIdentityPermission 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 [Granting CredentialAccessPermission 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,permClass='oracle.wsm.security.WSIdentityPermission',permTarget='resource=usermessagingserver,mapName=application.specific.map',permActions='getKey')
```

For more information about this WLST command, see "Infrastructure Security Custom WLST Commands" in the *WLST Command Reference for WebLogic Server*.

7.3.2.4 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 `WSIdentityPermission` 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 Configure the Credential Store"](#)

7.3.3 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 [Predefined Assertion Templates for Oracle Web Services](#)

- `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

7.4 Understanding Service Identity Certificate Extensions

For web services that implement a message-protection policy, the base64-encoded public certificate for the web service is published in the WSDL. The certificate is included for message protection policies regardless of whether the policy encrypts or decrypts data.

The certificate published in the WSDL is the service's public key by default, specified by the Encryption Key you configured in the keystore as described in [Overview of Configuring Keystores for Message Protection](#).

 **Note:**

In prior releases of OWSM, 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 so, you either identified the `keystore.recipient.alias` property on the Configuration page for the client policy, or specified a configuration override for the property on a per-client basis when you attached the policy (or programmatically).

If the public key 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.

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 verify the hostname, OWSM 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.

OWM provides domain configuration properties that enable you to specify whether to enforce web service policies by publishing the X509 certificate in the WSDL and whether to use the hostname verification feature. For details about setting these properties, see [Configuring Identity Extension Properties Using Fusion Middleware Control](#).

This section contains the following topics:

- [Ignoring the Service Identity Certificate Extension From the Client](#)
- [Ignoring Hostname Verification from the Client](#)

7.4.1 Ignoring the Service Identity Certificate Extension From the Client

Learn how to set the value of the **Ignore Identity WSDL** property.

 **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 **Ignore Identity WSDL** property is read automatically and no additional configuration is required. To turn identity verification on and off, set this property in as described in [Configuring Identity Extension Properties Using Fusion Middleware Control](#).

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");
```

7.4.2 Ignoring Hostname Verification from the Client

For a Java EE client, the value of the **Ignore Hostname Verification** property is read automatically and no additional configuration is required.

To turn hostname verification on and off, set this property as described in [Configuring Identity Extension Properties Using Fusion Middleware Control](#).

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");
```

7.5 Caching the Nonce with Oracle Coherence

To protect against replay attacks, several 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.

For example, see [oracle/wss10_username_token_with_message_protection_client_policy](#).

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. OWSM uses the Oracle Coherence cache to cache the nonce.

The application servers (using OWSM) must be part of the Coherence cluster. By default, when you run Configuration Wizard all the Managed Servers or clusters are part of the Coherence cluster. Within the cluster, two storage-enabled servers are sufficient for caching the nonce and provide failover protection. Those two servers can be the application servers, or, if the application server(s) are storage-disabled, two storage-enabled cache servers may be used.

Note:

You might need to configure the Coherence Cluster settings to avoid conflicts between them. For more information, see "Configuring and Managing Coherence Clusters" in *Administering Clusters for Oracle WebLogic Server*.

Depending on the topology of your environment, use one of the following procedures to enable Oracle Coherence to cache the nonce:

- [Caching the Nonce Where There Are No Managed Coherence Servers](#)

- [Caching the Nonce for Storage-Disabled WebLogic Servers and Storage-Enabled Managed Coherence Servers](#)

For information about setting the nonce time-to-live duration in the cache, see "Configuring Security Policy Enforcement Using Fusion Middleware Control" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

7.5.1 Caching the Nonce Where There Are No Managed Coherence Servers

Learn how to cache the coherence cluster topology.

Topics:

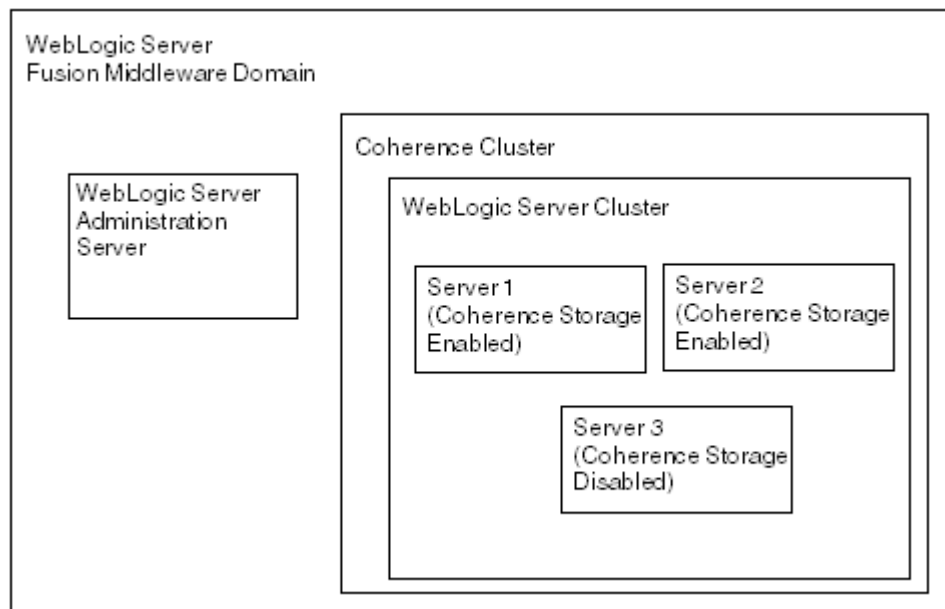
- [Understanding Coherence Cluster Topology Where There Are No Managed Coherence Servers](#)
- [Configuring the Standard Topology Using Fusion Middleware Configuration Wizard](#)

7.5.1.1 Understanding Coherence Cluster Topology Where There Are No Managed Coherence Servers

The Coherence cluster topology is formed by one or more WebLogic Server clusters or servers. Some of the servers are one or more storage-disabled (the WebLogic Server Coherence client tier) and one or more storage-enabled servers (the WebLogic Server Coherence server tier). There are no managed Coherence containers in this topology. This topology is typically used when the data storage requirement is low (usually less than 100 megabytes).

[Figure 7-9](#) illustrates a possible topology that consists of storage-enabled and storage-disabled servers within a WebLogic Server cluster. Notice that two servers are storage-enabled: in case of failover, one server can be used as backup for the other. When you run configuration wizard, all of the managed servers or clusters are part of the Coherence cluster by default.

Figure 7-9 Cluster Topology with Storage-Enabled and -Disabled WebLogic Servers



7.5.1.2 Configuring the Standard Topology Using Fusion Middleware Configuration Wizard

Follow the instructions in Installing the Infrastructure Software to install WebLogic Server and configure the standard installation topology. That standard topology provides a default coherence cluster that meets these requirements.

Note that in the **Create Domain** page in the Configuration Wizard, you can select **Create a new expanded domain**.

No other configuration is necessary. When you employ the `wss_username_token_client_policy` and `wss_username_token_service_policy` policies, the nonce will be stored in the Coherence cache.

7.5.2 Caching the Nonce for Storage-Disabled WebLogic Servers and Storage-Enabled Managed Coherence Servers

Learn how to cache the coherence cluster topology.

Topics:

- [Understanding Coherence Cluster Topology For Storage-Disabled WebLogic Servers and Storage-Enabled Managed Coherence Servers](#)
- [Configuring the Cluster Topology Using Fusion Middleware Configuration Wizard](#)

7.5.2.1 Understanding Coherence Cluster Topology For Storage-Disabled WebLogic Servers and Storage-Enabled Managed Coherence Servers

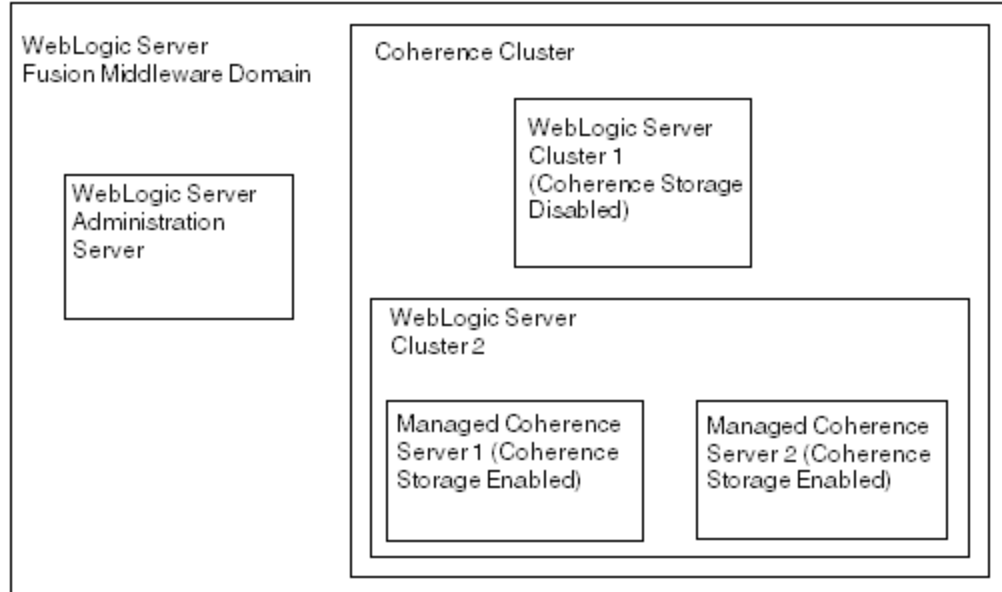
This Coherence cluster topology is formed by one or more storage-disabled WebLogic Server clusters or servers, and two or more managed storage-enabled Coherence servers (for failover protection). [Figure 7-10](#) illustrates this topology, which is typically used when the caching requirement is high (greater than 100 megabytes) and the WebLogic Servers are not used for data caching. Note that in this topology, every access to the cache requires a network round trip to the cache server.

Note:

The data storage requirement is a general recommendation from Oracle Coherence from a caching perspective. For OWSM, this requirement is very low from a caching perspective. If you have already enabled caching on at least two application servers (for failover protection), then it is optional to add more Coherence containers to the server group `WSM-CACHE-SVR`.

For example, if you think that the nonce will be lost if all of your application servers go down, then you can add additional Coherence containers as backup.

Figure 7-10 Cluster Topology with Storage-Enabled Managed Coherence Servers within the Coherence Cluster



7.5.2.2 Configuring the Cluster Topology Using Fusion Middleware Configuration Wizard

You can use Fusion Middleware Configuration Wizard to add the managed Coherence servers to a server group for caching the nonce. Then use WebLogic Server Remote console to ensure that local storage is enabled for the managed Coherence servers and that storage is disabled for the WebLogic servers.

 **Note:**

If you have Coherence storage enabled on the application servers, then the following steps are optional. They are required only if you want to backup the nonce on the managed Coherence containers.

1. Follow the instructions in Installing the Infrastructure Software to install WebLogic Server and configure the standard installation topology. See the following sections:
 - "Planning your Oracle Fusion Middleware Infrastructure installation"
 - "Installing the Oracle Fusion Middleware Infrastructure Software"
 - "Configuring Your Oracle Fusion Middleware Infrastructure Domain"

Note that in the **Create Domain** page in the Configuration Wizard, you can select **Create a new expanded domain**.

2. In the Managed Server page of the Configuration Wizard, identify the Coherence managed servers. Open the **Server Group** drop down list for the Coherence managed servers and choose **WSM-CACHE_SVR**.

- Once the domain has been created, use the WebLogic Server Remote Console to ensure that local storage is enabled for the managed Coherence servers and that storage is disabled for the WebLogic servers.

In the WebLogic Server Remote Console, you enable or disable storage for a server (or cluster) by selecting the **Local Storage Enabled** option in the **Coherence** subtab of the **Configuration** tab for the server (or cluster).

Note that if the WebLogic Cluster that is a member of the Coherence cluster is configured as storage-disabled, there is no need to mark the server as storage-enabled. (The storage-enabled setting is in the **Coherence** subtab for the WebLogic cluster and WebLogic server).

When you employ the `wss_username_token_client_policy` and `wss_username_token_service_policy` policies, the nonce will be stored in the Coherence cache.

7.6 About Configuring Partial Encryption with Fusion Middleware Control

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.

Topics:

- [Configuring Partial Encryption Using Fusion Middleware Control](#)
- [Securing SwA Attachments](#)

7.6.1 Configuring Partial Encryption Using Fusion Middleware Control

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.

In this example, a part of the SOAP message is encrypted using Fusion Middleware Control:

- Create a simple web service that approves a credit card number (`cardNr`). The next example shows a sample payload.

```
<soapenv:Body wsu:Id="Body-2grWlpYwjwsoskbLuMJZzg22"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-ws
security-utility-1.0.xsd">
```

```
  <aaav:validateTheCard xmlns:aaav="http://aaav:validatecred/">
    <aaav:cardNr>string</aaav:cardNr>
    <aaav:firstName>string</aaav:firstName>
    <aaav:lastName>string</aaav:lastName>
    <aaav:validUntilDate>string</aaav:validUntilDate>
  </aaav:validateTheCard>
```

```
</soapenv:Body>
```

- 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 7-11).
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`

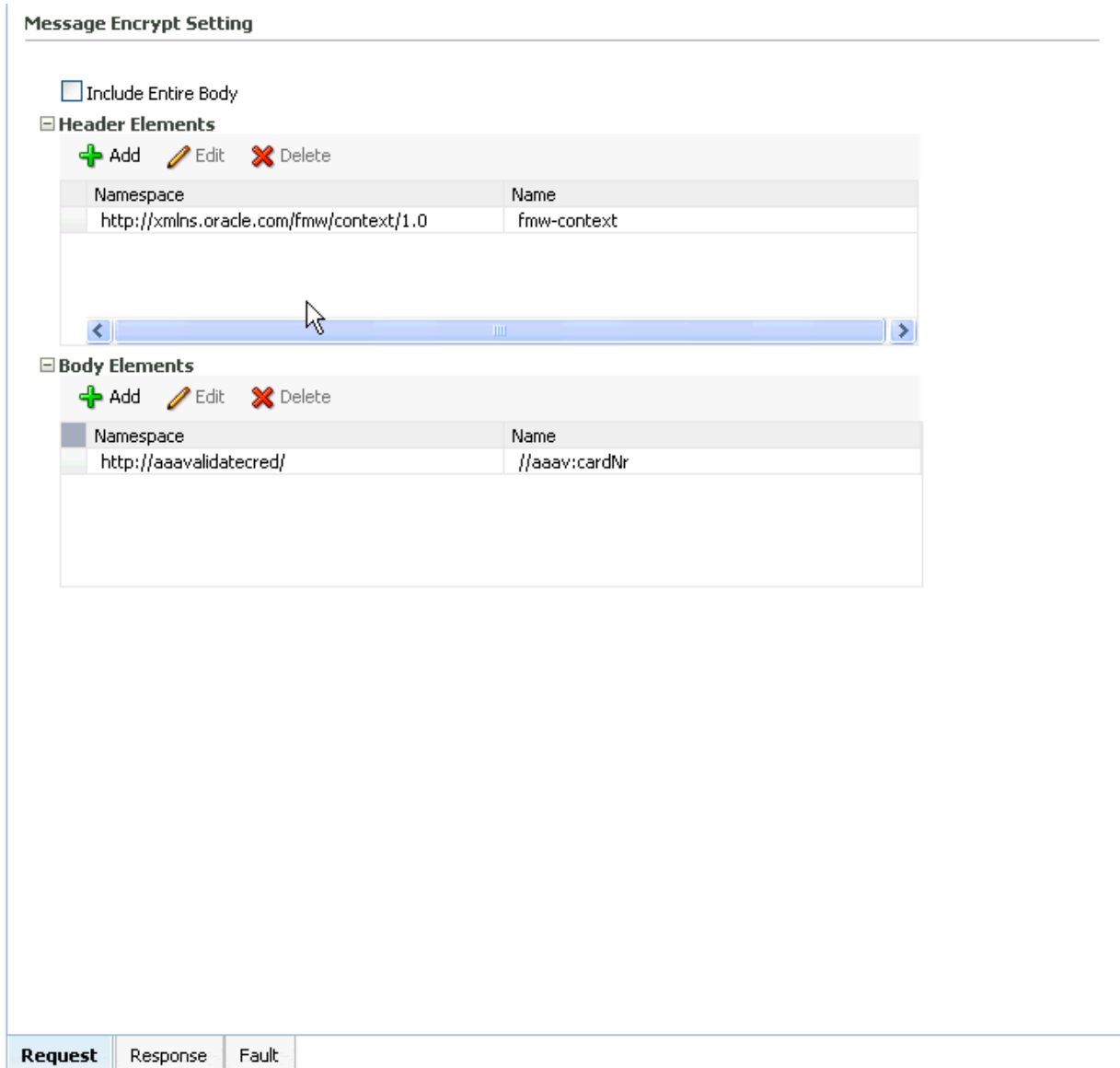
For more information on other fields in the Edit Policy page, see the table in [Message Signing and Encryption Settings for Request, Response, and Fault Messages](#).

The following example shows a 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 7-11 Example of Partial Encryption of Message Protection Policies



7.6.2 Securing SwA Attachments

Packaging SOAP messages with attachments (SwA) 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.

8

Protecting Personally Identifiable Information

Learn more on how to protect Personally Identifiable Information (PII) when outside the control of a security policy. PII refers to Social Security numbers, addresses, bank account numbers, and other similar information that is typically associated with one specific user and must generally be protected.

Topics:

- [Main Steps in Protecting PII Information](#)
- [Overriding the pii_security_policy Attributes Using WLST](#)
- [Decrypting PII Using API](#)

8.1 Main Steps in Protecting PII Information

This topic describes how to protect PII with the `oracle/pii_security_policy` policy.

Topics:

- [Approach to Follow to Determine What PII Data to Protect](#)
- [Composing the XPath Expressions to Protect the PII Data](#)
- [Configuring the PII Encryption Key](#)
- [Attaching the pii_security_policy Policy](#)
- [Attaching the pii_security_policy to SOA Composite](#)
- [Attaching the pii_security_policy to JCA Binding](#)



Note:

Configure the PII encryption key, then choose where you need to attach the `oracle/pii_security_policy` policy as appropriate for your environment.

8.1.1 Approach to Follow to Determine What PII Data to Protect

Examine the SOAP request and response messages, or the WSDL, and determine what PII data you want to protect.

There are two approaches:

- Deploy the SOA application and use JDeveloper (or another mechanism) to look at the SOAP messages and determine what you need to protect.
See the SOAP message example in "PII Policy XPath Expressions".
- Deploy the SOA application and look at the WSDL of the deployed application to determine what you need to protect.

You can display the WSDL document for the web service endpoint as described in "Viewing the Web Service WSDL Document" in *Administering Web Services with Oracle Fusion Middleware*.

You need to look at what data is being passed during both the request and response phases. That is, you may need to protect different data during the request and response.

8.1.2 Composing the XPath Expressions to Protect the PII Data

Compose the XPath expressions to protect the PII data in both the request and response messages.

See the SOAP message example in "PII Policy XPath Expressions" in *Understanding Oracle Web Services Manager* for guidance.

You later specify these XPath expressions in the `oracle/pii_security_policy` in the `request.xpath` and `response.xpath` attributes.

8.1.3 Configuring the PII Encryption Key

Configure a password CSF key to be used for generating the PII encryption key.

For more information, see [Adding Keys and User Credentials to Configure the Credential Store](#). The PII encryption key is derived from this password credential.

By default, `oracle/pii_security_policy` expects a key value of `pii-csf-key`, but you can change this.

If the web service client and the web service do not share a single credential store, then the PII encryption key must be present and identical in their respective credential stores.

8.1.4 Attaching the `pii_security_policy` Policy

Make a copy of the preconfigured `oracle/pii_security_policy` and then attach the copy to your web service and client.

Perform the following steps:

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure PII. Select the domain.
2. In the content pane, click **WebLogic Domain**, then **Web Services**, and then **Policies**.
3. Select the `oracle/pii_security_policy` policy and make a copy.
4. Examine the Salt, Iteration, and Key Size settings to make sure the defaults are acceptable, and change as needed. See [Table 18-108](#) for a description of these settings.
5. Edit the `request.xpath`s, `response.xpath`s and other `oracle/pii_security_policy` configuration properties of the copy. See "Understanding the PII Security Policy" in *Understanding Oracle Web Services Manager* to specify the XPath, namespaces, `csf.key`, and so forth.

Remember that you may need to protect different data during the request and response.

You may prefer to skip this step and instead override the attributes as described in [Step 7](#).

6. Attach the policy at design time or post-deployment, as described in [Attaching Policies to Manage and Secure Web Services](#).

In addition to the steps described in [Attaching Policies to Manage and Secure Web Services](#), the following sections provided here include use case-specific information:

- [Attaching the pii_security_policy to SOA Composite](#)
 - [Attaching the pii_security_policy to JCA Binding](#)
7. Optionally, use Fusion Middleware Control, WLST, or JDeveloper (or another mechanism) to override `request.xpath`s and other `oracle/pii_security_policy` attributes.

See "Understanding the PII Security Policy" to specify the XPath, namespaces, `csf.key`, and so forth.

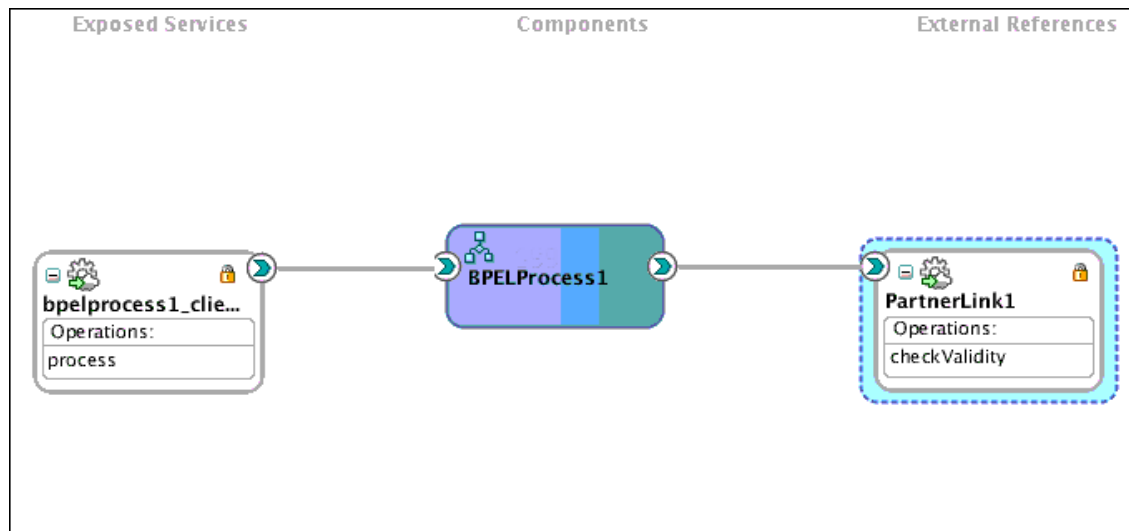
See [Overriding Policy Configuration Properties](#) for information on overriding configuration properties.

8.1.5 Attaching the pii_security_policy to SOA Composite

You must attach the policy only at the service/reference level, and then to both the client and web service.

Consider the SOA composite represented in JDeveloper shown in [Figure 8-1](#). For this composite, you would attach the policy to both the `bpelprocess_1client_ep` client and the `PartnerLink1` reference. You cannot attach the policy to a component.

Figure 8-1 Where to Attach the pii_security_policy for a SOA Composite



Perform the following steps to view the SOA composite and attach the `oracle/pii_security_policy` policy.

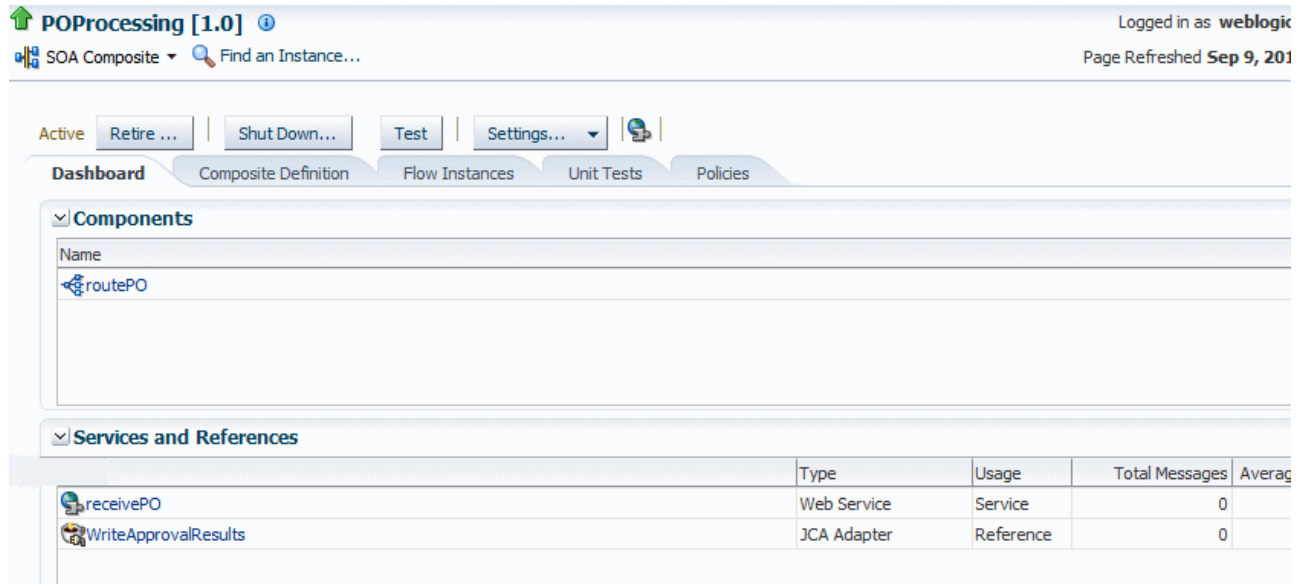
1. In the navigator, expand **SOA deployments**.
2. Select **soa-infra**, expand the SOA partition (for example, the default partition) and select the target SOA composite application.

The SOA composite home page displays.

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

The Components 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 8-2](#).

Figure 8-2 SOA Composite Application Dashboard Page



See "Viewing the Web Services and References in a SOA Composite" in *Administering Web Services* for additional SOA-specific information.

4. Attach a copy of the `oracle/pii_security_policy` policy to **both** the web service client and the reference binding.

 **Note:**

PII data requires that the composite have both entry and exit points: PII data is encrypted before entry and decrypted before exit.

For example, in [Figure 8-2](#) you would attach the copy to both `routePO` service and to the `WriteApprovalResults` reference.

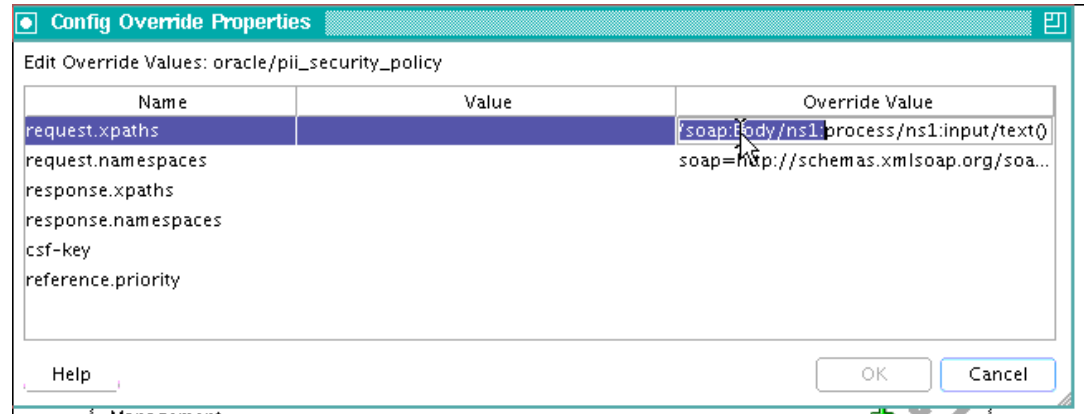
See [Attaching Policies to Manage and Secure Web Services](#) for information on attaching policies. Also see "Attaching Policies to Binding Components and Service Components" in *Developing SOA Applications with Oracle SOA Suite* for additional SOA-specific information.

5. Examine the Salt, Iteration, and Key Size settings to make sure the defaults are acceptable, and change as needed. See [Table 18-108](#) for a description of these settings.
6. Use Fusion Middleware Control, WLST, or JDeveloper (or another mechanism) to override `request.xpath`s, `response.xpath`s and other attributes.

The PII data for the request and response may be different.

[Figure 8-3](#) shows a sample JDeveloper screen.

Figure 8-3 Overriding request.xpathns Attribute



Do this for both the web service and client, and ensure that they match.

See "Understanding the PII Security Policy" in *Understanding Oracle Web Services Manager* to specify the XPathns, namespaces, csf.key, and so forth.

See [Overriding Policy Configuration Properties](#) for information on overriding configuration policies.

8.1.6 Attaching the pii_security_policy to Oracle Service Bus

You can attach the PII policy to JCA adapters for both SOA and Oracle Service Bus.

See "Managing Service and Reference Binding Components" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite* for information on attaching the PII policy to JCA bindings.

See "Hiding Personally Identifiable Information in Messages" in *Developing Services with Oracle Service Bus* for information on how to attach `oracle/pii_security_policy` to Oracle Service Bus.

8.1.7 Attaching the pii_security_policy to JCA Binding

You can attach the PII policy to JCA adapters for both SOA and Oracle Service Bus.

See "Managing Service and Reference Binding Components" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite* for information on attaching the PII policy to JCA bindings.

See "Hiding Personally Identifiable Information in Messages" in *Developing Services with Oracle Service Bus* for information on how to attach `oracle/pii_security_policy` to Oracle Service Bus.

8.2 Overriding the pii_security_policy Attributes Using WLST

You can override the `oracle/pii_security_policy` attributes using WLST.

Perform the following steps to override the `oracle/pii_security_policy` attributes using WLST:

1. Connect to the running instance of WebLogic Server, as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services with Oracle Fusion Middleware*. Begin a session using the `beginWSMSession` command, as described in [Attaching Policies Directly Using WLST](#). For example:

```
wls:/wls_domain/serverConfig> beginWSMSession()
```

```
Session started for modification.
```

2. Select the policy subject that you want to work with. See [Identifying and Selecting the Policy Subject Using WLST](#).

Remember that you can attach `oracle/pii_security_policy` only in specific cases, as described in "When Can You Use the PII Policy" in *Understanding Oracle Web Services Manager*. If you attach the policy to a non-supported location, WLST generates validation errors.

3. Use the `attachWSMPolicy` command to attach a copy of the `oracle/pii_security_policy` to a web service port, as described in [Attaching Policies Directly Using WLST](#). For example:

```
wls:/base_domain/serverConfig> attachWSMPolicy('oracle/pii_security_policy')
```

```
Policy reference "oracle/pii_security_policy" added.
```

4. Override the `oracle/pii_security_policy` policy attributes. For example, the following command sets the `request.xpathns` attribute:

```
setWSMPolicyOverride('oracle/pii_security_policy','request.xpathns','//ns1:SSN' )
```

Similarly, the following command unsets the `request.xpathns` attribute:

```
setWSMPolicyOverride('oracle/pii_security_policy','request.xpathns' )
```

5. Write the contents of the current session to the repository using the `commitWSMSession()` command.

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

8.3 Decrypting PII Using API

It is possible that the encrypted PII might be needed for making some decision. In this case, you must explicitly decrypt the PII with the provided API before it can be used.

For example, assume that a credit card number is marked as PII and is encrypted at the SOA service binding (entry point). If the credit card number is required inside a BPEL process to determine the type of credit card, then you must decrypt the credit card number using the API.

The `oracle.security.xmlsec.pii.PIIISecurity.java` class is provided for this purpose. This class has the following method:

```
Class: oracle.security.xmlsec.pii.PIIISecurity.java
```

```
/**
 * Converts cipher text string to plain text using password based key
 * derivation algorithm (PBKDF2).
 *
 * @param ciphertext text to decrypt
 * @param password password for key derivation
 * @param pbkdfAlgo key derivation algorithm which should be PBKDF2
 * @param pbkdfSalt non-null and non-empty salt for key derivation
```

```
* @param pbkdfIteration iteration count for key derivation
* @param keySize size of key for key derivation
* @param encAlg data encryption algorithm. it should be in the form:
*           "algorithm/mode/padding" for ex. AES/CBC/PKCS5Padding
* @return plain text
*/
public static String decrypt(String ciphertext, char password[], String
    pbkdfAlgo, String pbkdfSalt, int bkdfIteration, int keySize, String encAlg);
```

The `decrypt` method returns the decrypted value of the encrypted PII, but does not update the actual value, which remains encrypted.

For SOA, you can invoke this API using the Java Embedding feature, which is described in "Using Java Embedding in a BPEL Process in Oracle JDeveloper" in *Developing SOA Applications with Oracle SOA Suite*.

**Note:**

Do not log the decrypted value of the PII unless you are completely aware of the security ramifications of doing so.

9

Configuring Transport-Level Security (SSL)

You can configure Secure Socket Layer (SSL), a transport-level security protocol. SSL can be either one-way or two-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. With two-way SSL, the server presents a certificate to the client and the client presents a certificate to the server. To use SSL, you must set up keystores and truststores in your environment.

For more information on SSL, see [Understanding Transport-level and Application-level Security](#) in Understanding Oracle Web Services Manager.

Topic:

- [About Configuring Keystores for SSL](#)
- [Configuring One-Way SSL on WebLogic Server](#)
- [Configuring Two-Way SSL on WebLogic Server](#)
- [Configuring One-Way SSL for a Web Service Client](#)
- [Configuring Two-Way SSL for a Web Service Client](#)
- [Understanding SSL Configuration on Oracle HTTP Server](#)

9.1 About Configuring Keystores for SSL

You can use SSL with the OWSM Policy Manager by configuring keystores.

The policies listed in [OWSM Policies that Require You to Configure SSL](#) or [List of Policies That Require You to Configure Two-Way 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 client, and optionally a server, 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 the following topics:

- [Understanding KSS Keystore Configuration on WebLogic Server](#)
- [Configuring a JKS Keystore on WebLogic Server](#)
- [Configuring Synchronization of JKS Keystore File on Cluster](#)

9.1.1 Understanding KSS Keystore Configuration on WebLogic Server

The OPSS Keystore Service provides an alternate mechanism to manage keys and certificates for message security. This section briefly summarizes the steps that are required to configure the OPSS Keystore Service in WebLogic Server.

As described in "Managing Keys and Certificates with the Keystore Service" in *Securing Applications with Oracle Platform Security Services*, You use the OPSS Keystore Service to create and maintain keystores of type KSS.

See the following two sources for complete information:

- "Servers: Configuration: Keystores" in Oracle WebLogic Server Administration Console Online Help.
- *Configuring Keystores* in *Administering Security for Oracle WebLogic Server 14c (14.1.2)*.

This section describes the following topics:

- [Configuring the OPSS Keystore Service for Demo Identity and Trust](#)
- [Recreating the OPSS Keystore Service for Demo Identity and Trust](#)
- [Configuring the OPSS Keystore Service for Custom Identity and Trust](#)

9.1.1.1 Configuring the OPSS Keystore Service for Demo Identity and Trust

The KSS demo identity and demo trust keystores are preconfigured when you create a domain, and no additional configuration of these keystores is required.

Perform the following steps to configure the OPSS Keystore Service for demo identity and trust:

1. From the WebLogic Server Remote Console, navigate to the Domain -> Security -> Advanced page, and verify that the "Use KSS For Demo" check box is enabled.
2. Configure the WebLogic Server instance to use Demo Identity and Demo Trust, as described in [Configure keystores](#).
3. Configure SSL for the WebLogic Server instance, as described in [Configuring One-Way SSL on WebLogic Server](#) and [Configuring Two-Way SSL on WebLogic Server](#).

Remember that the WebLogic Server DefaultHostnameVerifier has been modified to accept the non-standard `DemoCertFor_<WLS Domain Name>` hostname format. Other hostname verifiers may not support this format.

4. Restart WebLogic Server.

9.1.1.2 Recreating the OPSS Keystore Service for Demo Identity and Trust

The KSS demo identity keystore is preconfigured when you create a domain, and no additional configuration of this keystore is required.

However, in case you have subsequently changed or removed the KSS demo identity keystore, use the instructions in this section to recreate the keystore.

You can perform the OPSS Keystore Service operations using either Fusion Middleware Control or the Keystore Service commands with WLST. This section demonstrates the Fusion Middleware Control steps, but "Managing Keys and Certificates with the Keystore Service" describes both options.

Perform the following steps to configure an OPSS Keystore Service for demo identity and trust:

1. Launch Fusion Middleware Control.
2. From the **WebLogic Domain** menu, select **Security** then **Keystore**.
3. Create a keystore named `demoidentity` in the `system` stripe. (See "Creating a Keystore with Fusion Middleware Control" for more information.)

- a. Select the `system` stripe and click **Create Keystore**.

The Create Keystore page is shown in [Figure 9-1](#).

Figure 9-1 Create Keystore

- b. Name this keystore `demoidentity`.
 - c. Set the protection type to **Password**.
 - d. Set the password to `DemoIdentityKeyStorePassPhrase`, and confirm.
 - e. Clear the **Grant Permission** check box.
 - f. Do not specify a code base URL.
4. Select the `demoidentity` keystore you just created and click **Manage**.

Enter the `DemoIdentityKeyStorePassPhrase` password.

The Manage Certificates screen shown in [Figure 9-2](#) appears.

Figure 9-2 Manage Certificates

5. Click **Generate Keypair** to generate a private/public key pair.

The Generate Keypair screen is shown in [Figure 9-3](#).

Figure 9-3 Generate Keypair

- a. Specify `DemoIdentity` as the alias for the key pair.
- b. Specify the Common Name as `DemoCertFor_<WLS Domain Name>`, where `DemoCertFor_` is a required constant and `<WLS Domain Name>` is the WebLogic Server domain name. For Example: `DemoCertFor_base_domain`.

 **Note:**

The WebLogic Server `DefaultHostnameVerifier` has been modified to accept this non-standard hostname format when you set the "Use KSS For Demo" flag in the security configuration for the Weblogic Server domain. Other hostname verifiers may not support this format.

- c. Specify other site-specific information as appropriate.
 - d. You can accept the default RSA key size if appropriate for your environment. Oracle requires a key length of 1024 bits or larger.
 - e. Specify the password as `DemoIdentityPassPhrase`.
 - f. Click **OK**.
6. From the WebLogic Server Remote Console, navigate to the Domain -> Security -> Advanced page, and enable the "Use KSS For Demo" check box.
 7. Configure the WebLogic Server instance to use Demo Identity and Demo Trust, as described in Configure keystores.
 8. Configure SSL for the WebLogic Server instance, as described in [Configuring One-Way SSL on WebLogic Server](#) and [Configuring Two-Way SSL on WebLogic Server](#).

Remember that the WebLogic Server DefaultHostnameVerifier has been modified to accept the non-standard `DemoCertFor_<WLS Domain Name>` hostname format. Other hostname verifiers may not support this format.

9. Restart WebLogic Server.

9.1.1.3 Configuring the OPSS Keystore Service for Custom Identity and Trust

You must configure the OPSS Keystore Service before you can use it for custom identity and trust with WebLogic Server.

You can perform the OPSS Keystore Service operations using either Fusion Middleware Control or the Keystore Service commands with WLST. This section demonstrates the Fusion Middleware Control steps, but "Managing Keys and Certificates with the Keystore Service" describes both options.

Perform the following steps to configure an OPSS Keystore Service for custom identity and trust:

1. Launch Fusion Middleware Control.
2. From the **WebLogic Domain** menu, select **Security** then **Keystore**.
3. Create a keystore in the `system` stripe. (See "Creating a Keystore with Fusion Middleware Control" for more information.)
 - a. Select the `system` stripe and click **Create Keystore**.

The Create Keystore page is shown in [Figure 9-4](#).

Figure 9-4 Create Keystore

The screenshot shows a "Create Keystore" dialog box with the following fields and options:

- Keystore Stripe:** system
- Keystore Name:** (empty text box)
- Protection:** Policy (radio button), Password (radio button, selected)
- Keystore Password:** (empty text box)
- Confirm Password:** (empty text box)
- Grant Permission:**
- Code Base URL:** (empty text box)
- Buttons:** OK, Cancel

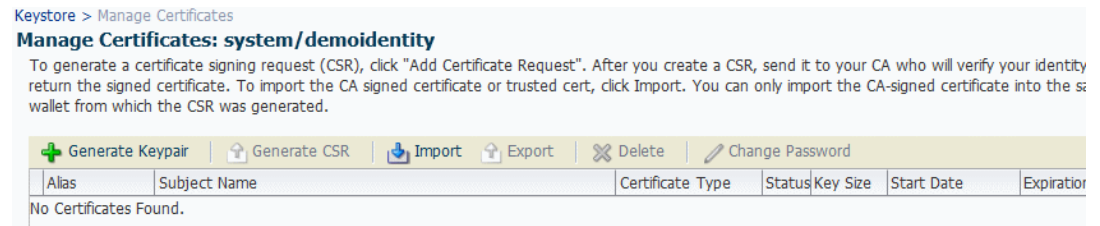
- b. Name this keystore.
- c. Set the protection type to Password.
- d. Set the password.
- e. Clear the Grant Permission check box.
- f. Do not specify a code base URL.

4. Select the keystore you just created and click **Manage**.

Enter the password.

The Manage Certificates screen shown in [Figure 9-5](#) appears.

Figure 9-5 Manage Certificates



5. Click **Generate Keypair** to generate a private/public key pair.

The Generate Keypair screen is shown in [Figure 9-6](#).

Figure 9-6 Generate Keypair

- a. Specify the alias for the key pair.
- b. Specify 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. Specify the password.
- e. Click **OK**.

6. You have the option to use this KSS Demo CA-signed key pair as-is, or to obtain a signed certificate from a reputable vendor such as Entrust, Verisign, and so forth.

To obtain the signed certificate from a reputable vendor, select the alias for the key pair and click **Generate CSR**. After you create a CSR, send it to your CA, which will authenticate the certificate request and create a digital certificate based on the request.

See "Importing a Certificate with Fusion Middleware Control" in *Importing a Certificate with Fusion Middleware Control* for instructions on how to import the CA-signed certificate.

7. If you do not use the preconfigured OPSS Keystore Service trust store `kss://system/trust`, you must create your own.

 **Note:**

Oracle recommends you use the OPSS Keystore Service trust store because it simplifies trust configuration.

To create your own trust store, create another OPSS Keystore Service keystore, and import trusted certificates. See *Importing a Certificate with Fusion Middleware Control in Securing Applications with Oracle Platform Security Services* for instructions on how to import trusted certificates.

8. Configure the WebLogic Server instance to use KSS for Custom Identity and Trust, as described in *Configure keystores*. You specify the fully-qualified path to the key store as the URI in the form `kss://system/keystore-name`. The keystore type is KSS.
9. Configure SSL for the WebLogic Server instance, as described in [Configuring One-Way SSL on WebLogic Server](#) and [Configuring Two-Way SSL on WebLogic Server](#).

All the server SSL attributes are dynamic; when modified via the Console, they cause the corresponding SSL server or channel SSL server to restart and use the new settings for new connections. Old connections will continue to run with the old configuration. To ensure that all the SSL connections exist according to the specified configuration, you must reboot WebLogic Server.

9.1.2 Configuring a JKS Keystore on WebLogic Server

You can configure the JKS keystore in WebLogic Server.

This topic briefly summarizes the steps that are required to configure the JKS keystore in WebLogic Server.

You may find that using the KSS Keystore Service as described in [Understanding KSS Keystore Configuration on WebLogic Server](#) is the most convenient method of configuring a keystore. However, you also have the option of using a JKS keystore, including the WebLogic default identity keystore `Demoidentity.jks` and the default trust keystore `DemoTrust.jks`.

For more information see :

- Servers: Configuration: Keystores in Oracle WebLogic Server Administration Console Online Help.
- *Configuring Keystores* in *Administering Security for Oracle WebLogic Server 14c (14.1.2)*.

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 trusted 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://docs.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 `DOMAIN_HOME\security` and `WL_HOME\server\lib` directories respectively, 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.

9.1.3 Configuring Synchronization of JKS Keystore File on Cluster

You can configuring JKS keystore in fresh installs of 12.2.1 or using JKS in upgrade scenarios

If you want synchronization of JKS keystore file to happen on cluster without server restart, 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/
application-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.

9.2 Configuring One-Way SSL on WebLogic Server

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 [About Configuring Keystores for SSL](#), 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 [Configuring SSL](#).

To configure SSL:

1. In the left pane of the WebLogic Server Remote 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.
 - Builtin SSL Validation and Cert Path Validators: Uses the built-in trusted CA-based validation and uses configured CertPathValidator providers to perform extra validation.

9.3 Configuring Two-Way SSL on WebLogic Server

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 [About Configuring Keystores for SSL](#), you can configure its two-way SSL attributes if the policy or template you are using requires it, as described in [List of Policies That Require You to Configure Two-Way SSL](#).

This section summarizes the steps required to configure SSL on WebLogic Server. For complete information, see [Configuring SSL](#).

Perform the following steps to configure two-way SSL:

1. In the left pane of the WebLogic Server Remote 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 Requested and Enforced.
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.
 - Builtin SSL Validation and Cert Path Validators: Uses the built-in trusted CA-based validation and uses configured CertPathValidator providers to perform extra validation.

9.4 Configuring One-Way 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:

- Ensure that WebLogic Server obtains a digital certificate that the client automatically trusts, because it has been issued by a trusted certificate authority.
- Create a certificate registry that lists all the individual certificates trusted by WebLogic Server, and then ensure that the client trusts these registered certificates.
- If the client and WebLogic Server are on the same system, have them use the same trust store.

To configure one-way SSL for a web services client, you set the following properties in the client's JVM:

- `javax.net.ssl.trustStore` -- The system property that specifies the trust store location. For JKS it is the name of the file that contains the trust store. For KSS it is the KSS URI.

You can either use the existing JKS or KSS trust store, or create a new one. For example, you can have `javax.net.ssl.trustStore` point to the default `kss://system/trust` trust store.

To create a new JKS trust store you can use the `keytool` utility. To create a new KSS keystore you can use Fusion Middleware Control.

- `javax.net.ssl.trustStoreType` -- The type of trust store object that you want the default `TrustManager` to use, either JKS or KSS.
- `javax.net.ssl.trustStorePassword` -- JKS only. Specifies the trust store's password.

For Oracle Infrastructure web service clients, you can also set individual properties in their descriptor files. The properties set in individual descriptor files override properties set using the system properties:

- For a SOA composite, set the properties in `composite.xml`.
- For ADF web service data control (ADFDC), set the properties in `connections.xml`.
- For ADF business components and WebCenter clients, set the properties in `oracle-webservices-client.xml`.

9.5 Configuring Two-Way SSL for a Web Service Client

To configure two-way SSL for a Web Service Client, 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.

Note:

See "Configuring SOA Composite Applications for Two-Way SSL Communication" in *Administering 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.

Do one of the following to configure Two-Way SSL for a Web Service Client:

- Ensure that the client application obtains a digital certificate that WebLogic Server automatically trusts, because it has been issued by a trusted certificate authority.
- 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.
- If the client and WebLogic Server are on the same system, have them use the same key store and trust store.

To configure SSL for a web service client, make sure that the following properties are set in the client's JVM:

- `javax.net.ssl.keyStore` -- The system property that specifies the keystore location. For JKS it is the name of the file that contains the key store. For KSS it is the KSS URI.

You can either use the existing JKS or KSS key store, or create a new one. For example, you can have `javax.net.ssl.keyStore` point to the default `kss://system/castore` key store so that the web service client uses the X.509 certificates that WebLogic Server uses.

To create a new JKS key store you can use the `keytool` utility. To create a new KSS keystore you can use Fusion Middleware Control.

- `javax.net.ssl.keyStoreType` -- The type of KeyStore object that you want the default TrustManager to use, either JKS or KSS.
- `javax.net.ssl.trustStore` -- The system property that specifies the trust store location. For JKS it is the name of the file that contains the trust store. For KSS it is the KSS URI.

You can either use the existing JKS or KSS trust store, or create a new one. For example, you can have `javax.net.ssl.trustStore` point to the default `kss://system/trust` trust store so that the web service client trusts the X.509 certificates that WebLogic Server trusts.

To create a new JKS trust store you can use the `keytool` utility. To create a new KSS keystore you can use Fusion Middleware Control.

- `javax.net.ssl.trustStoreType` -- The type of TrustStore object that you want the default TrustManager to use, either JKS or KSS.
- `javax.net.ssl.trustStorePassword` -- JKS only. Specifies the trust store's password.
- `HTTPClient.ssl.identityAlias` -- The alias of the certificate you want to use for identity. If the alias is null, the JSSE provider chooses one of the aliases in the keystore.

For Oracle Infrastructure web service clients, you can also set individual properties in their descriptor files. The properties set in individual descriptor files override properties set using the system properties:

- For a SOA composite, set the properties in `composite.xml`.
- For ADF web service data control (ADFDC), set the properties in `connections.xml`.
- For ADF business components and WebCenter clients, set the properties in `oracle-webservices-client.xml`.

9.6 Understanding SSL Configuration 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 OWSM 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.

For more information, see:

- "Configuring SSL in Oracle Fusion Middleware", in *Administering Oracle Fusion Middleware*.
- "Configuring SSL" in *Administering Security for Oracle WebLogic Server 14c (14.1.2)*.
- "Set Up SSL" in the *Oracle WebLogic Server Administration Console Online Help*
- "Configuring Secure Sockets Layer" in *Administrator's Guide for Oracle HTTP Server*.

This section describes how to configure the policies that require one-way SSL and two-way SSL. It includes the following topics:

- [Configuring One-Way SSL on Oracle HTTP Server](#)
- [Configuring Two-Way SSL on Oracle HTTP Server](#)

9.6.1 Configuring One-Way SSL on Oracle HTTP Server

This section describes how to configure one-way SSL on Oracle HTTP Server.

For more information on the OWSM policies that require one-way SSL configuration, see [OWSM Policies that Require You to Configure SSL](#).

Perform the following steps to use one-way SSL:

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 the following example.

```
# 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 is 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 *Administering Oracle Fusion Middleware*.
- d. Restart Oracle HTTP Server.

For more information, see "Configuring SSL in Oracle Fusion Middleware" in *Administering Oracle Fusion Middleware*.

2. Create a wallet as described at "Managing Keystores, Wallets, and Certificates" in *Administering Oracle Fusion Middleware* and replace the default wallet. The default wallet is located in the `ORACLE_INSTANCE/config/OHS/<ohs_name>/keystores/default` directory. The following example shows sample commands to create a wallet 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=owsm,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 Remote 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 Online 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 shown in the following example.

```

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:\\OWSM_
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);
}

```

9.6.2 Configuring Two-Way SSL on Oracle HTTP Server

This section describes how to configure two-way SSL on Oracle HTTP Server.

For more information on the OWSM policies that require two-way SSL configuration, see [List of Policies That Require You to Configure Two-Way SSL](#).

Perform the following sections to use two-way SSL:

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 the following example.

```
# 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 is 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 *Administering Oracle Fusion Middleware*.

- d.** Restart Oracle HTTP Server.

For more information, see "Configuring SSL in Oracle Fusion Middleware" in *Administering Oracle Fusion Middleware*.

- 2.** Create a wallet as described at "Managing Keystores, Wallets, and Certificates" in *Administering Oracle Fusion Middleware* and replace the default wallet. The default wallet is located in the `ORACLE_INSTANCE/config/OHS/<ohs_name>/keystores/default` directory. See the following example for sample commands to create a wallet 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 Remote 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 Online 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 the following example.

```
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");
    ((BindingProvider) class1).getRequestContext().put(BindingProvider.PASSWORD_
PROPERTY, "password");
    System.setProperty("javax.net.ssl.trustStore", "D:\\OWSM_
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:\\OWSM_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

Configuring Authorization Using Oracle Web Services Manager

You can configure authorization using OWSM. Authorization (also referred to as access control) enables you to determine what operations authenticated clients can access.

Topics:

- [Overview of Authorization](#)
- [Determining Which Resources to Protect](#)
- [Determining Authorization Permissions](#)
- [Determining the OPSS Resource Name](#)
- [About Configuring Fine-Grained Authorization Using Oracle Entitlements Server](#)
- [Configuring the Oracle HTTP Server to Specify the Request Origin](#)
- [Using OAuth 2.0 with Oracle Web Services Manager](#)
- [Understanding OWSM integration with 3rd Party Servers](#)

10.1 Overview of Authorization

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 template.

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 to access the web service.

For summaries of the authorization policies available in the current release, see "[Authorization Policies](#)" in [Determining Which Predefined Policies to Use for a Web Service](#). [Oracle Web Services Manager Predefined Policies](#), summarizes the security policies that enforce authorization, and indicates whether the policy is enforced at the transport layer or SOAP header.

For more information about authorization, see "Understanding Authorization" in *Understanding Oracle Web Services Manager*.



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.

10.2 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.

Note:

The predefined policies are read-only and cannot be edited. You can, however, create new policies using the predefined policies as a base. For information about creating a new policy, see [Creating and Editing Web Service Policies](#). Once you have created the new policy, you can edit the policy and modify the settings or set the configuration properties as desired.

- **Constraint Match** -- 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 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 the Oracle HTTP Server to Specify the Request Origin](#).

Note the following:

- The **Constraint Match** 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 Match** -- 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 Match are determined by the web service methods. For example, if the web service method is `validate(amountAvailable)`, enter the Action Pattern as `validate`.

- **Resource Match** -- 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 Match, the policy is enforced only for those web services that match the criteria. That is, entering a specific value in the Resource Match field limits the scope of the authorization policy. The default of `*` protects all resources (namespace of web service + web service name) of the bulk-attached web services.

- **Permission Class** -- By default, it is `oracle.wsm.security.WSFunctionPermission`. The class must be in the classpath.
- **Roles** -- 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

10.3 Determining Authorization Permissions

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 Match**, and **Action Match** parameters on the **Assertions** details page for the authorization policy determine what resources are being protected by the policy, as shown in [Figure 10-1](#).

The `oracle/binding_permission_authorization_policy` also contains **resource** and **action** configuration properties that you can use to set a different action and resource. If you set these properties, or override them, the new values are used in the attached web service instead of the action and resource you configure using the **Resource Match** and **Action Match** parameters. For details about using configuration overrides, see [Overriding Policy Configuration Properties](#).

You have the option to change the **Permission Class** setting 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://docs.oracle.com/javase/7/docs/api/java/security/Permission.html>. The default is `oracle.wsm.security.WSFunctionPermission`.

Figure 10-1 The Permission Settings for a Policy

Details

▼ **Authorization Permission**

Guard

Action Match

Constraint Match

Resource Match

Check Permission

Permission Class

- The OPSS Application Policies page specifies whether the authenticated subject has invoke access to the **Resource Name** listed there, as shown in [Figure 10-2](#).

Figure 10-2 Adding a Permission on the OPSS Create Application Grant Page

Customize resource or actions for selected permission.

Customize

Permission Class

Resource Name

Permission Actions

OPSS uses the authorization policy's **Assertions** details 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 10-1](#) and [Figure 10-2](#).

On the **Assertions** details page for the authorization policy, assume that you specify the following to protect the `validate` method of the `http://project11/CreditValidation` web service:

```
Action Match:      validate
Resource Match:    http://project11/CreditValidation
Permission 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.

10.4 Determining the OPSS Resource Name

the resource target of the `WSFunctionPermission` is enhanced to include the actual web service operation name. The OPSS resource name can include the operation name.



Note:

The OPSS resource name can include the 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`.

10.5 About Configuring 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 end-to-end across the enterprise. OES is integrated with OWSM, and you can use OES together with OWSM for fine-grained authorization.

As described in "Understanding Oracle Entitlements Server Integration" in *Understanding Oracle Web Services Manager*, 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 OWSM, and you can use OES together with OWSM for fine-grained authorization.

"Understanding Oracle Entitlements Server Integration" 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 OWSM 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](#)
- [Understanding Attributes for Obligations](#)
- [About Configuring OES Policies For Fine-Grained Authorization](#)
- [About Configuring OES Policies For Coarse-Grained Authorization](#)
- [About Configuring OES Policy For Masking](#)
- [Understanding How to Attach OWSM OES Policy](#)

10.5.1 Prerequisites for Configuring OES Integration

In addition to your OWSM 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 OWSM.

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. For more information see "[Installing and Configuring Oracle Entitlements Server](#)" in *Installation Guide for Oracle Identity and Access Management* for installation information.

10.5.2 Understanding Attributes for Obligations

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.

As described in "How Attributes Are Processed", 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 "Viewing the Web Service WSDL Document" in *Administering Web Services with Oracle Fusion Middleware*.

10.5.3 About Configuring OES Policies For Fine-Grained Authorization

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

As described in "OES Integration: The Big Picture" in *Understanding Oracle Web Services Manager*, 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 OWSM policy, as described in [Understanding How to Attach OWSM OES Policy](#). 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:

- [Configuring the OES Resource for Masking](#)
- [Creating Authorization Policy to Return Obligations](#)
- [Creating Actual OES Authorization Policy for Coarse-Grained Authorization](#)

10.5.3.1 Configuring the OES Resource for Masking

You must map the OES resource name to the OWSM resource name. When making an authorization call from OWSM, the resource name is passed to OES, and this name must exactly match the one defined in the OES policy.

As described in "Resource Mapping and Naming", you must map the OES resource name to the OWSM resource name. When making an authorization call from OWSM, 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 10-3 Create OES Application

The screenshot shows the 'General' tab of the Oracle Entitlements Server console. There are three tabs: 'General', 'Delegated Administrators', and 'Policy Distribution'. The 'General' tab is active. Below the tabs, there are three input fields: 'Display Name' with the value 'HelloWorldServiceEar', '* Name' with the value 'HelloWorldServiceEar', and 'Description' which is empty. A tooltip labeled 'Description' is positioned to the right of the empty 'Description' field.

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 OWSM 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.

10.5.3.2 Creating Authorization Policy to Return Obligations

This procedure helps you to create authorization policy to Return Obligations.

To create authorization policy to Return Obligations, follow these steps:

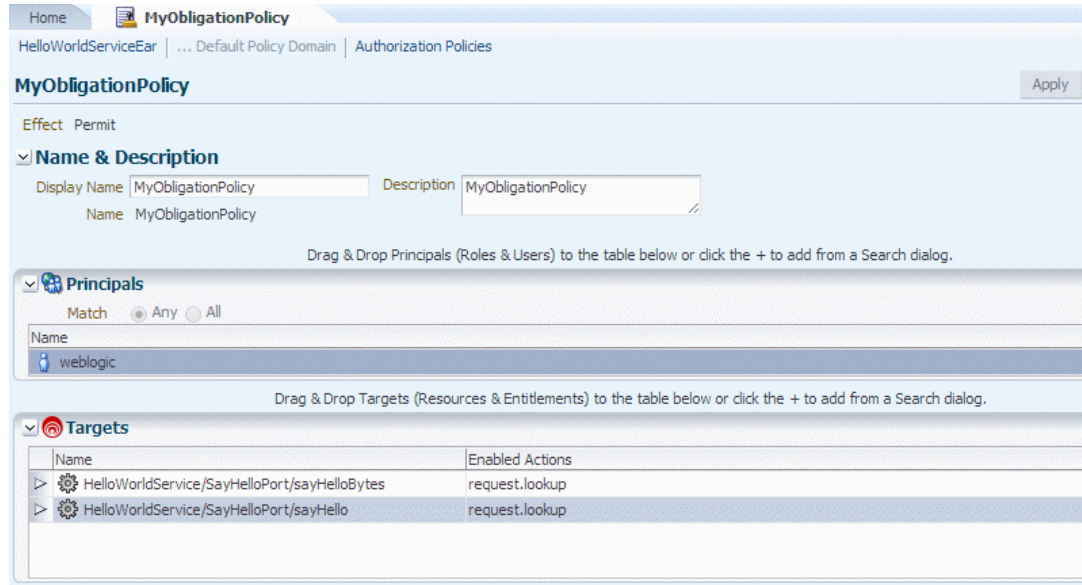
1. Create a new OES authorization policy for the application you created in ["Configuring the OES Resource for Masking"](#).

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 10-4](#) shows an example screen.

Figure 10-4 Adding Authorization Policy for Obligations



2. From the Obligations tab, add Obligations.

As described in "How Attributes Are Processed", 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 "Attribute Types Supported for OES Policies" in *Understanding Oracle Web Services Manager*.

For the purpose of example, consider how [Figure 10-5](#) is derived from and reflects the sample obligations shown in the following table in this section.

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:

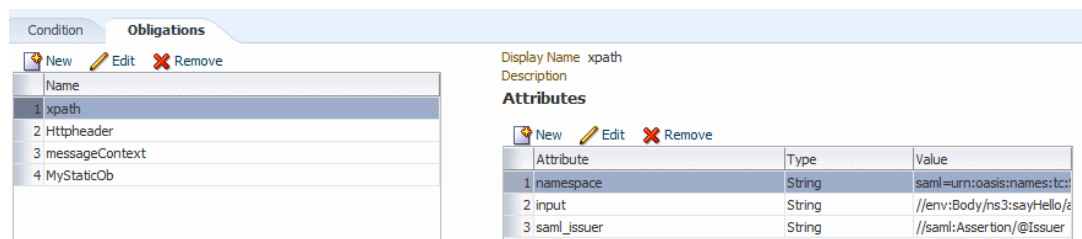
```
<attribute_type>:<obligation_name>
```

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.

Obligation Name	Attribute Name	Attribute Value
xpath	input	//env:Envelope//env:Body/ns3:sayHello/arg0/text()
xpath	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=.
xpath	saml_issuer	//saml:Assertion/@Issuer
HTTPHeader	proxy_auth	Proxy-Authorization
HTTPHeader	authHeader	Authorization
messageContext	authMethod	oracle.wsm.internal.authentication.method
messageContext	endpoint	oracle.j2ee.ws.runtime.endpoint-url
MyStaticOb	org	oracle
MyStaticOb	country	US

Figure 10-5 Sample Obligations For Authorization Policy



The screenshot shows the OES console interface. On the left, under the 'Obligations' tab, there is a list of obligations: 1 xpath, 2 Httpheader, 3 messageContext, and 4 MyStaticOb. The 'xpath' obligation is selected. On the right, the 'Attributes' section is visible, showing a table with the following data:

Attribute	Type	Value
1 namespace	String	saml=urn:oasis:names:tc:
2 input	String	//env:Body/ns3:sayHello/e
3 saml_issuer	String	//saml:Assertion/@Issuer

10.5.3.3 Creating Actual OES Authorization Policy for Coarse-Grained Authorization

Create the OES authorization policy to perform the actual authorization.

This policy uses the Obligations provided by OWSM to make the real authorization decision.

In addition, OWSM passes to OES the authenticated subject, the target resource and requested action, as well as a set of implicit attributes (as described in "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 ["Creating Authorization Policy to Return Obligations"](#).

For example, to match the Obligations you created in [Creating Authorization Policy to Return Obligations](#) you would need to add the following attributes:

- saml_issuer
- input
- authHeader
- endpoint
- country

Note:

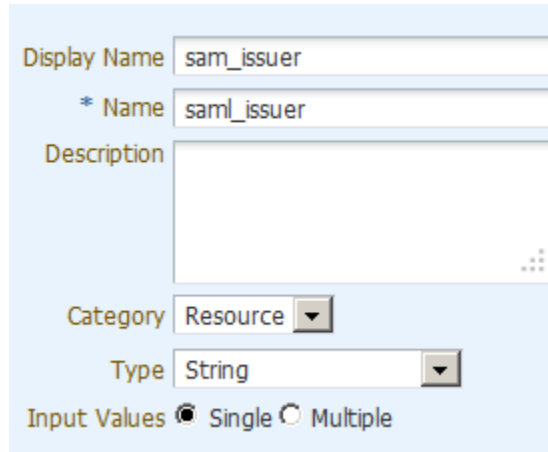
If you plan to use any of the implicit attributes (as described in "Attribute Types Supported for OES Policies"), you must add them as well.

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 10-6 Create New Attributes



Display Name

* Name

Description

Category

Type

Input Values Single Multiple

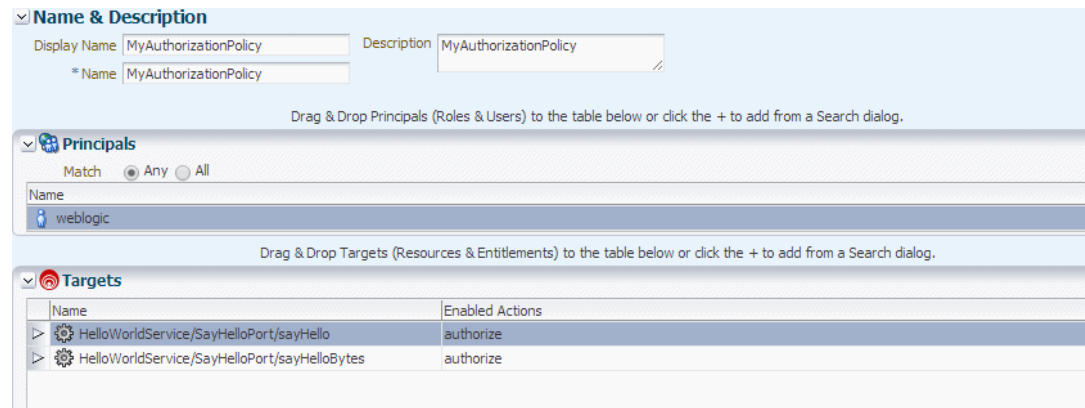
2. Add all of the new attributes to your resource type.
3. Create a new OES authorization policy for the application you created in [Configuring the OES Resource for Masking](#).

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 10-7](#) 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 10-7 Adding Actual Authorization Policy



Name & Description

Display Name Description

* Name

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 10-8](#).

Figure 10-8 Create the Condition


```

Condition
  Obligations
  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 "OWSM" at //env:Body/ns3:sayHello/arg0.
 - Other conditions are met.
5. Attach the OWSM OES policy, as described in [Understanding How to Attach OWSM OES Policy](#).

10.5.4 About Configuring OES Policies For Coarse-Grained Authorization

You can configure OES policies for Coarse-Grained Authorization.

As described in "OES Integration: The Big Picture" in *Understanding Oracle Web Services Manager*, 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 OWSM policy, as described in [Understanding How to Attach OWSM OES Policy](#). However, you need to decide on the method you plan to use so that you can configure the OES authorization policy accordingly.

Topics:

- [Configuring the OES Resource for Coarse-Grained Authorization](#)
- [Creating Actual OES Authorization Policy for Fine-Grained Authorization](#)

10.5.4.1 Configuring the OES Resource for Coarse-Grained Authorization

As described in "Resource Mapping and Naming", you must map the OES resource name to the OWSM resource name. When making an authorization call from OWSM, the resource name is passed to OES, and this name must exactly match the one defined in the OES policy.

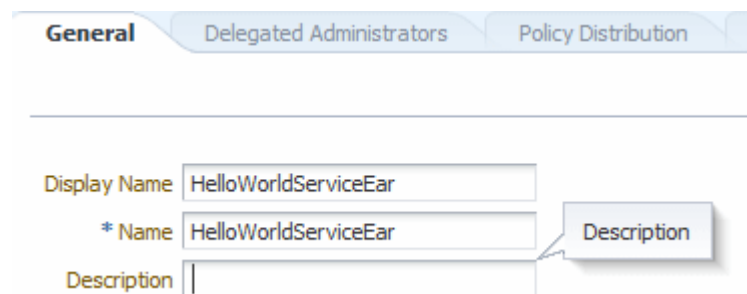
For the purpose of this example, assume that you have a deployed SOAP web service with the following characteristics:

- 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 10-9 Create OES Application



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 OWSM 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.

10.5.4.2 Creating Actual OES Authorization Policy for Fine-Grained Authorization

Create the OES authorization policy to perform the actual authorization.

OWSM passes to OES the authenticated subject, the target resource and requested action, as well as a set of implicit attributes (as described in "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 "Attribute Types Supported for OES Policies"), you must add them.

 **Note:**

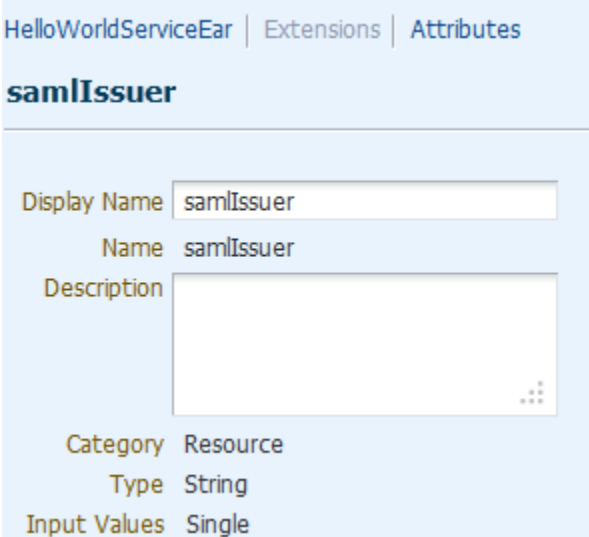
All Attributes and Functions (both custom and predefined) are created, collected and further managed under the Extensions node of the Application. For more information, see "[Managing Attributes and Functions as Extensions](#)" in *Administrator's Guide for Oracle Entitlements Server*.

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 10-10](#).

Figure 10-10 Add Implicit Attributes Needed for Conditions



>HelloWorldServiceEar | Extensions | Attributes

samlIssuer

Display Name

Name

Description

Category	Resource
Type	String
Input Values	Single

3. Create a new OES authorization policy for the application you created in [Configuring the OES Resource for Coarse-Grained Authorization](#).

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 10-11](#) shows an example screen.

Figure 10-11 Adding Actual Authorization Policy

The screenshot shows the configuration interface for adding an actual authorization policy. It is divided into three main sections:

- Name & Description:** Contains input fields for 'Display Name' (MyAuthorizationPolicy), 'Description' (MyAuthorizationPolicy), and '* Name' (MyAuthorizationPolicy).
- Principals:** Includes a 'Match' section with radio buttons for 'Any' (selected) and 'All'. Below is a table with one entry:

Name
weblogic
- Targets:** Includes a table with two entries:

Name	Enabled Actions
HelloWorldService/SayHelloPort/sayHello	authorize
HelloWorldService/SayHelloPort/sayHelloBytes	authorize

4. Attach the OWSM OES policy, as described in [Understanding How to Attach OWSM OES Policy](#).

10.5.5 About Configuring OES Policy For Masking

You can configure the OES policy for masking.

Topics:

- [Configuring the OES Resource](#)
- [Creating Masking Policy to Return Obligations](#)
- [Creating Actual OES Masking Policy](#)

10.5.5.1 Configuring the OES Resource

As described in "Resource Mapping and Naming", you must map the OES resource name to the OWSM resource name. When making a masking call from OWSM, 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 10-12 Create OES Application

The screenshot shows a web interface with three tabs: 'General', 'Delegated Administrators', and 'Policy Distribution'. The 'General' tab is active. Below the tabs, there are three input fields: 'Display Name' with the value 'HelloWorldServiceEar', '* Name' with the value 'HelloWorldServiceEar', and 'Description' which is empty. A callout box labeled 'Description' points to the empty 'Description' field.

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.

10.5.5.2 Creating Masking Policy to Return Obligations

Create a masking policy to Return Obligations by performing the following steps:

1. Create a new OES policy for the application you created in [Configuring the OES Resource](#).
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", 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, OWSM 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.) OWSM uses the result of this query to call the actual masking policy described in [Creating Actual OES Masking Policy](#).

You use the attribute names from this Obligation when you create the Obligation for the actual masking policy.

10.5.5.3 Creating Actual OES Masking Policy

Create the OES authorization policy to perform the actual masking.

This policy uses the Obligations provided by OWSM 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 "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 [Configuring the OES Resource](#).

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. OWSM 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 OWSM OES policy, as described in [Understanding How to Attach OWSM OES Policy](#).

10.5.6 Understanding How to Attach OWSM OES Policy

You can attach OWSM OES policy.

Topics:

- [Attaching the OWSM OES Policy](#)
- [Configuration Properties and Overrides](#)

10.5.6.1 Attaching the OWSM 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 [Attaching Policies to Manage and Secure Web Services](#).

Attach the OWSM `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](#) for a description of the attributes you can override.

See [Overriding Policy Configuration Properties](#) for information on overriding configuration policies.

7. Also attach any of the authentication policies at design time or post-deployment, as described in [Attaching Policies to Manage and Secure Web Services](#).

10.5.6.2 Configuration Properties and Overrides

You can set the configuration properties shown in [Table 10-1](#) 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 10-1 OWSM 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 <code>\${}</code> notation.
resource.type	Resource type defined in OES. Value can be static or dynamic that uses <code>\${}</code> notation. <ul style="list-style-type: none"> For SOAP application, must be <code>WS_SERVICE</code>. For SOA component, must be <code>COMPONENT</code>.
resource.name	Resource name defined in OES. Value can be static or dynamic that uses <code>\${}</code> notation. <ul style="list-style-type: none"> For SOAP and SOA reference, must be of the form <code>web-service-name/port/web service operation</code>. For SOA component, must be of the form <code>SOA component name/web service operation</code>.
lookup.action	Action that will be used during attributes lookup. Can be <code>request.lookup</code> or <code>response.lookup</code> . Value can be static or dynamic that uses <code>\${}</code> notation.
execute.action	Action that will be used during real authorization or masking. Default values are <code>authorize</code> for authorization and <code>mask</code> for masking use case. Value can be static or dynamic that uses <code>\${}</code> 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

10.6 Configuring the Oracle HTTP Server to Specify the Request Origin

You can configure the Oracle HTTP Server to specify the request origin.

The **Constraint Match** 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 configure the Oracle HTTP Server to specify the request origin, 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.

10.7 Using OAuth 2.0 with Oracle Web Services Manager

You can use the Oracle Mobile and Social OAuth2 authorization framework with Oracle Web Services Manager.

Topics:

- [About OAuth2 with Oracle Web Services Manager](#)
- [Configuring OAuth2 for Use With Oracle Web Services Manager Policies](#)
- [About OAuth Based Anonymous User Authentication](#)
- [About Multitenant Support for OAuth Token Validation](#)
- [About OAuth Based Anonymous User Authentication](#)
- [About Multitenant Support for OAuth Token Validation](#)

- [About Proxy Configuration for Outbound OAuth 2.0 on the Client Application](#)
- [About Support for OAuth 2.0 Tokens with Application Defined Scopes](#)



Note:

This section assumes that you are familiar with both the terminology and the conceptual and configuration information described in the following sections of *Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*:

- [Understanding OAuth Services](#)
- [Configuring OAuth Services](#)

10.7.1 About OAuth2 with Oracle Web Services Manager

OAuth2 support in Oracle Web Services Manager is based on "The OAuth 2.0 Authorization Framework" specification.

Which is available at <http://tools.ietf.org/html/rfc6749>.

Oracle Web Services Manager uses the Oracle Mobile and Social OAuth2 service as the authorization server for the OAuth2.0 protocol interactions. For more information, see [Understanding Mobile and Social](#) in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

Oracle Web Services Manager 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.

- [Understanding 2-legged OAuth2](#)
- [Supported Authorization Grant Types in 2-Legged Authorization](#)
- [How Client Credentials Are Determined in 2-Legged Authorization](#)
- [Relationship of User Credentials, Client Credentials, and Subject in 2-Legged Authorization](#)

10.7.1.1 Understanding 2-legged OAuth2

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 Web Services Manager 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 Web Services Manager JWT service policies to the web service. The Oracle Web Services Manager server-side agent validates the access token.

- `oracle/http_jwt_token_service_policy`
- `oracle/http_jwt_token_over_ssl_service_policy`
- `oracle/multi_token_rest_service_policy`
- `oracle/multi_token_over_ssl_rest_service_policy`
- `oracle/wss11_saml_or_username_token_with_message_protection_service_policy`
- `oracle/http_oauth2_token_client_policy`
- `oracle/http_oauth2_token_opc_oauth2_client_policy`
- `oracle/http_oauth2_token_identity_switch_opc_oauth2_over_ssl_client_policy`
- `oracle/http_oauth2_token_opc_oauth2_over_ssl_client_policy`
- `oracle/http_oauth2_token_identity_switch_over_ssl_client_policy`
- `oracle/http_oauth2_token_over_ssl_client_policy`
- `oracle/http_oauth2_config_client_policy`

10.7.1.2 Supported Authorization Grant Types in 2-Legged Authorization

As previously described, an authorization grant is a credential representing the resource owner's authorization. Oracle Web Services Manager 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 [Configuring OAuth2 for Use With Oracle Web Services Manager Policies](#)

- **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 `federated.client.token` property in the attached `oauth2` client policy to specify the user name and password to use.

- **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 Web Services Manager generates the JWT token locally based on client credentials stored in the Oracle Web Services Manager 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.

JWT token is generated for the client using the values of the `oauth2.client.csf.key` and `keystore.sig.csf.key` properties. Ensure that the property `federated.client.token` is set to `true`.

- **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 `oauth2.client.csf.key` property in the attached `oauth2` client policy 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](#).

10.7.1.3 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.

10.7.1.4 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

<code>subject.precedence</code>	<code>csf-key</code>	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	N/A	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 <code>csf-key</code> entry.	The user name from the <code>csf-key</code> /user name is configured.

10.7.2 About OWSM OAuth2 Client API

OAuth2 Client API feature allows to get access token without the need of creating any rest client, it enables consumer of API to fetch access token and use it in any outbound request. This OAuth2 access token can then be used by the API caller to call a service in the OAuth2 resource server protected by the OAuth2 Auth Server.

OWSM does the following:

- Invoke OAuth2 server to authenticate and authorize to obtain Access Token.
- Secure the messages to the Resource server with the Access Token.

OWSM cannot do either of the above functions independently. This OAuth2 Client API feature can be used to Invoke OAuth2 server to authenticate and authorize to obtain Access Token.

This section includes the following topic:

- [Configuring OWSM OAuth2 Client Credentials](#)

10.7.2.1 Configuring OWSM OAuth2 Client Credentials

OWSM supports for OAuth2 on the client side as an integrated solution for the whole process. This OAuth2 Client API feature enables calls to be made independently to the OAuth2 server.

The process to configure OAuth2 client credentials and invoke the client app to get the Access Token consists of the following steps:

1. You must register your client under the register Client Section of the Oauth server and update the following fields:
 - Name - Select a Client name which is different from the client ID.
 - Accessible Resource - Select appropriate resources which are accessible to this client.
 - Trusted client - You can ignore this check for the time being. You can click on it later and upload a certificate once your client side configurations are done.
 - After providing all the necessary values, click on **Register**.
2. Attach client policies which will be used with OAuth 2 Client API.

 **Note:**

OWSM policies can be attached using LPA or GPA. It is recommended that the oauth2 config policy should be attached as GPA and http_oauth2_token_over_ssl_client_policy attached as LPA.

3. You must Configure OAuth2 client credentials in client domain.
 - a. Click on Show Secret to get the client Id and secret of registered client with OAuth2 server.

 **Note:**

The OAuth 2 client must be a trusted client to work with OWSM OAuth 2 Client API.

- b. Now connect to client domain and execute the following WLST to configure client credentials in CSF store:

```
createCred(map="oracle.wsm.security", key="<replace with OAuth2 server>",
user="<replace with the OAuth 2 client id>", password="<replace with OAuth2
```

4. You must Setup KeyStore at Client Domain.

Note:

In 14c, KSS is the default keystore which is used.

- a. Connect to the client domain and execute the following WLST :

```
svc = getOpssService(name='KeyStoreService')

svc.createKeyStore(appStripe='owsm', name='keystore',
password='',permission=true)
```

- b. Generate keypair:

```
svc.generateKeyPair(appStripe='owsm', name='keystore', password='', dn='CN=OWSM,
OU=ST, O=Oracle, L=RedWood, ST=CA, C=US', keysize='2048', alias
```

Note:

Orakey is the default key alias, to sign the client request. The key will be used to sign the client and user assertion to be sent to the OAuth 2 server. As a trusted OAuth 2 client, the signing certificate must be uploaded to the OAuth 2 server.

5. Update client code to use OWSM OAuth2 client API.

This shows a sample servlet based application. Update your servlet client code as -

```
package client.jaxrs;
import java.io. IOException;
import java.io. PrintWriter;
import java.security. AccessController;
import java.util. Date;
import javax.security.auth. Subject;
import javax.servlet. ServletConfig;
import javax.servlet. ServletException;
import javax.servlet.http. HttpServlet;
import javax.servlet.http. HttpServletRequest;
import javax.servlet.http. HttpServletResponse;
import oracle.wsm.metadata.feature. PolicyReferenceFeature;
import oracle.wsm.metadata.feature. PolicySetFeature;
import oracle.wsm.metadata.feature. PropertyFeature ;
import oracle.wsm.security.oauth2. OAuth2ClientTokenManager;
import oracle.wsm.security.oauth2. OAuth2TokenContext;
import oracle.wsm.security.oauth2. OAuth2TokenResponse;
import oracle.wsm.security.util. SecurityConstants;
public class SampleClient extends HttpServlet {
public void init(ServletConfig config) throws ServletException {
super.init(config);
}
/**
* set OWSM Policy details
```

```

* @return
* @return
*/
private PolicySetFeature createOWSMPolicySetFeature() {
PropertyFeature tenantName new PropertyFeature (
SecurityConstants.ConfigOverride.CO_USER_TENANT_NAME, inoracleindiatrial52406")
PolicyReferenceFeature[] clientPRF = new PolicyReferenceFeature[] {
new PolicyReferenceFeature ("oracle/http_oauth2_token_over_ssl_client_policy",
new PropertyFeature[] { tenantName}) );
PolicySetFeature policySetFeature = new PolicySetFeature(clientPRF);
return policySetFeature;
}
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
try {
OAuth2ClientTokenManager oauth2ClientTokenManager =
OAuth2ClientTokenManager.getInstance ();
OAuth2TokenContext oauth2TokenContext =
OAuth2ClientTokenManager.getOAuth2TokenContext ();
PolicySetFeature policySetFeature = createOWSMPolicySetFeature ();
oauth2TokenContext.setPolicySetFeature(policySetFeature);
oauth2ClientTokenManager.getAccessToken(oauth2TokenContext);
OAuth2TokenResponse oauth2TokenResponse =
oauth2TokenContext.getOAuth2TokenResponse();
String accessToken = oauth2TokenResponse.getAccessToken ();
String tokenType = oauth2TokenResponse.getTokenType ();
long expiresIn = oauth2TokenResponse.getExpiresIn();
out.println ("<br><b> accessToken:<br><b>" + accessToken);
out.println ("<br><b> tokenType : <b>" + tokenType);
out.println ("<br><b> expiresInSec : <b>" + new Date(System.currentTimeMillis() +
expiresIn * 1000));
out.println ("<br>; ;-----request
processed-----");
} catch (Exception e) {
e.printStackTrace();
out.println("Exception is : " + e + "; exc message is: " + e.getMessage());
}
}
public void destroy () {
super.destroy ();
}
}

```

6. Provide oracle.wsm.security.WSIdentityPermission to your client application via WLST :

```
grantPermission(appStripe=None, codeBaseURL='file:${common.components.home}/modules/
oracle.wsm.common/wsm-agent-core.jar',principalClass=None
```

7. Connect to the client admin server and run the following WLST to export the client's signing certificate to a file:

```
svc.exportKeyStoreCertificate(appStripe='owsm', name='keystore', password='',
alias='orakey', keypassword='welcome1', type='Certificate',filepath
```

8. Configure OAuth server to trust client.

- a. Go to OAuth Configuration and modify the client.
- b. Trust the client and upload the certificate file generated in client environment.

9. Invoke client app to get Access Token.

- Deploy the client app and invoke the URL from browser.

 **Note:**

You can see the successful response along with the access token from OAuth2 server. This AT can be used to invoke the resource deployed on the resource server.

10.7.3 Configuring OAuth2 for Use With Oracle Web Services Manager Policies

As a prerequisite to configuring Mobile and Social OAuth2 for use with Oracle Web Services Manager, you should be familiar with the configuration information.

For more information, see *Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*.

For detailed information about configuring OAuth2 for use with Oracle Web Services Manager, see [Configuring OAuth Services](#) in *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

10.7.4 Enabling User Assertion by Username/Password in OAuth Policy

Each application is assigned a different generic user as there are multiple applications in a Weblogic Domain.

User assertion using unique username/password is enabled by attaching OAuth policies. Two new policies attached for user assertion are as follows:

- `http_oauth2_token_with_resource_owner_creds_client_policy`
- `http_oauth2_token_with_resource_owner_creds_over_ssl_client_policy`.

The client side OAuth policies - `grant_type` (default value/absence of it will mean `grant_type` is `client_credentials`) includes a new config override.

Add `grant_type` in the override prop. In these policies the `grant_type` will be set to `password`.

 **Note:**

If `grant_type` is `password` then `username` and `password` will be set in the header of the request to the oauth server using the `csf-key`.

The OAuth policies do the following:

- Assign Resource owner password credential with Basic auth flow
- Assign Resource owner password credential with JWT bearer flow

The OAuth policies are attached in the following ways:

- Attaching Policies and Providing Config Overrides Programmatically
- Attaching OAuth config policies from WLST
- Attaching OAuth config policies from EM

 **Note:**

The resource owner password credentials grant type is suitable in cases where the resource owner has a trust relationship with the client, such as the device operating system or a highly privileged application. The authorization server should take special care when enabling this grant type and only allow it when other flows are not viable.

10.7.5 About OAuth Based Anonymous User Authentication

OWSM supports OAuth based anonymous user access.

In order to access the REST APIs (anonymous or non-anonymous), the calls are made, such that an OAuth ClientId and Client Secret are generated. The anonymous users do not use their user credentials to get access to the REST endpoints. The anonymous APIs allow the applications to invoke the endpoints without requiring user authentication, while non-anonymous APIs require user authentication.

 **Note:**

Anonymous APIs are not open to the public but they require an access token in the Authorization header.

In order to support OAuth based anonymous user access, OWSM has a policy property `enable_anonymous_client_at`. This property establishes the subject as anonymous, even if the user does not exist in the Identity cloud service, when the `client_id` and the `sub` values in the token are the same.

This configure override, `enable_anonymous_client_at` uses client access token from identity cloud service, as anonymous token if the value is `true`. The default value of the policy is `false` and it is enabled by, setting this configuration to `true`. In scenarios where, the configuration is set to `true`, OWSM authenticates the user anonymously to create an anonymous subject after the identity cloud service client access token has been validated.

The following OWSM policies have the `enable_anonymous_client_at` property:

```
oracle/http_jwt_token_over_ssl_service_policy
oracle/http_jwt_token_service_policy
oracle/http_oam_token_service_policy
oracle/multi_token_cg_sso_over_ssl_rest_service_policy
oracle/multi_token_cg_sso_over_ssl_rest_service_policy
oracle/multi_token_cg_sso_rest_service_policy
oracle/multi_token_cg_sso_rest_service_policy
oracle/multi_token_over_ssl_rest_service_policy
oracle/multi_token_over_ssl_rest_service_policy
oracle/multi_token_rest_service_policy
oracle/multi_token_rest_service_policy
oracle/multi_token_sso_over_ssl_rest_service_policy
oracle/multi_token_sso_over_ssl_rest_service_policy
oracle/multi_token_sso_rest_service_policy
oracle/multi_token_sso_rest_service_policy
```

The JWT and OAM assertions inside the multi_token policies have `enable_anonymous_client_at` config override.

The following is an example of attaching the policy and changing configuration override using WLST:

```
beginWSSession()  
createWSPolicySet('rest-resource-policy-set', 'rest-resource', 'Domain("**")')  
attachWSPolicy('oracle/multi_token_rest_service_policy')  
commitWSSession()  
  
beginWSSession()  
selectWSPolicySet('rest-resource-policy-set')  
setPolicySetPolicyOverride('oracle/multi_token_rest_service_policy',  
'enable_anonymous_client_at', 'true')  
commitWSSession()
```

10.7.6 About Multitenant Support for OAuth Token Validation

OWSM validates the OAuth token, once the valid trust certificates are imported into the trust store.

In scenarios with multiple tenant support, OAuth token is validated, either by importing the trust certificates of all tenants (created or yet to be created) into the trust store or by using discovery feature.

The discovery feature helps OWSM to fetch the required trust certificates from Identity Cloud Service (IDCS) and use that to validate the OAuth tokens. The discovery feature is used to setup trust at global level using REST API, where all tenant will use the global information to set up trust.

The global discovery information in case of IDCS, does not have any tenant information and it is used to access the JWK/Discovery information for any tenant.

Note:

This requirement to support discovery feature for multiple tenants in a single instance environment is different from layered multitenant scenario where each tenant has its own database.

See Also:

- Manage Token Issuer Trust Configurations in *REST API for Managing Credentials and Keystores with Oracle Web Services Manager* for REST API methods to Import Global Discovery Configuration and Export TrustDocument Name Configuration.

10.7.7 About Proxy Configuration for Outbound OAuth 2.0 on the Client Application

You can configure the proxy for OAuth 2.0 server token endpoints on the client application using the WLST command or REST API. To avoid network errors while using the OAuth 2.0

with Oracle Web Services Manager, you must enable proxy for the OAuth 2.0 server token endpoints that requires proxy configuration.

Topics

- [Enabling Proxy for OAuth2 Server Token Endpoints](#)
- [Managing the Proxy Exclusion Lists](#)

10.7.7.1 Enabling Proxy for OAuth2 Server Token Endpoints

Use the WLST commands or REST API to enable proxy for a single or multiple OAuth 2.0 Server endpoints.

Topics

- [Enabling Proxy Using WLST](#)
- [Enabling Proxy Using REST API](#)

10.7.7.1.1 Enabling Proxy Using WLST

You can enable proxy for a single or multiple OAuth 2.0 Server endpoints (`token.uri`) using the WLST commands.

Complete the following procedure to enable a proxy for OAuth 2.0 server token endpoints:

1. Connect to the running instance of WebLogic Server as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Configures proxy for a URL:

```
setWSMTokenIssuerTrustProxy(issuer,identifier, proxyHost,proxyPort)
```

Argument	Value
issuer	Specify None.
identifier	Identifier which represents the URL. The following URL formats are supported: <ul style="list-style-type: none">• Absolute URI. For example: <code>https://accounts.example.com/o/oauth2/token</code>• Base URL ending with the * (wildcard). For example, <code>https://accounts.example.com/*</code>.• Specify * (wildcard) to enable proxy for all OAuth2 server token endpoints in a client application. For example, <code>"*"</code>.
proxyHost	Proxy host.
proxyPort	Proxy port.

See `setWSMTokenIssuerTrustProxy` in *WLST Command Reference for Infrastructure Components*.

The following examples set proxy for a specific and multiple token endpoints.

For a Specific Token Endpoint on the Client Application

```
setWSMTokenIssuerTrustProxy(None,"https://accounts.example.com/o/oauth2/token","www.proxy.com","80")
```

For all Token Endpoints on the Client Application

```
setWSMTokenIssuerTrustProxy(None,"*","www.proxy.com","80")
```


10.7.7.1.2 Enabling Proxy Using REST API

You can use the token issuer trust REST API to enable a proxy for a OAuth 2.0 server token endpoint.

Complete the following procedure to enable a proxy for a OAuth 2.0 server token endpoint (`token.uri`):

1. Use the GET method to view the existing token attribute rule for the OAuth 2.0 server token endpoint (`token.uri`).

```
curl -i -X GET -u username:password http://myhost:7001/wsm-pmrest/v2/trust/
{trustdocumentName}attributerule?issuename=name&identifier=(identifier)
```

The following table lists the query parameters:

Name	Description
issuename	Issuer name of the token attribute rule. Specify NONE.
identifier	Identifier which represents the encoded URL. For example: <code>https%3A%2F%2Faccounts.example.com%2Fo%2Foauth2%2Ftoken</code>

Example

```
curl -i -X POST -u Smith:Password -H Content-type:application/json http://
myhost:7001/wsm-pmrest/v2/trust/trustdocName/attributerule?
identifier=(https%3A%2F%2Faccounts.example.com%2Fo%2Foauth2%2Ftoken)
```

The following shows the example of the response body.

```
[
  {
    "identifier": "https%3A%2F%2Faccounts.example.com%2Fo%2Foauth2%2Ftoken"
  }
]
```

2. If attribute rule for the OAuth 2.0 server token endpoint (`token.uri`) does not exist then you must create an attribute rule using the POST method.

```
curl -i -X POST -u username:password http://host:port/wsm-pmrest/v2/trust/
{trustdocumentName}/attributerule
```

Example

```
curl -i -X POST -u Smith:Password -d @reset.json -H Content-type:application/json
http://myhost:7001/wsm-pmrest/v2/trust/trustdocName/attributerule/NONE/
https%3A%2F%2Faccounts.example.com%2Fo%2Foauth2%2Ftoken
```

The following shows an example of the response indicating the request succeeded.

```
{
  "STATUSCODE": "20101",
  "MESSAGE": "TokenAttributeRule
\"[https%3A%2F%2Faccounts.example.com%2Fo%2Foauth2%2Ftoken, issuename:null]\" are
successfully created."
}
```

The following shows the example of the response body.

```
[
  {
    "identifier":
    "https%3A%2F%2Faccounts.example.com%2Fo%2Foauth2%2Ftoken",
  }
]
```

3. Use the POST method to add proxy for the OAuth 2.0 server token endpoint (`token.uri`) attribute rules.

```
curl -i -X POST -u username:password http://host:port/wsm-pmrest/v2/trust/
{trustdocumentName}/attributerule/{IssuerName}/{Identifier}/proxy
```

The following table lists the query parameters:

Name	Description
issuename	Issuer name of the token attribute rule. Specify NONE.
identifier	Identifier which represents the Encoded URL. For example: https%3A%2F%2Faccounts.example.com%2Fo%2Foauth2%2Ftoke n

Example

```
curl -i -X POST -u Smith:Password -d @reset.json -H Content-type:application/json
http://myhost:7001/wsm-pmrest/v2/trust/trustdocName/attributerule/NONE/
http%3A%2F%2Fmyexample.com%2F%2A/proxy
```

The following shows an example of the response indicating the request succeeded.

```
{
  "STATUSCODE": "20171",
  "MESSAGE": "Proxy for TokenAttributeRule for issuer \"NONE\" and KeyIdentifier
  \"https%3A%2F%2Faccounts.example.com%2Fo%2Foauth2%2Ftoken\" is successfully created."
}
```

The following shows an example of the request body.

```
{
  "proxyhost": "myHostName",
  "proxyport": "1234"
}
```

10.7.7.2 Managing the Proxy Server Exclusion Lists

If the proxy is enabled for all the token endpoints on the client application then you can override the proxy for specific token endpoints by creating a proxy exclusion list. You can create a proxy exclude lists using either WLST or REST API.

Topics

- [Managing Proxy Exclusion List Using WLST](#)
- [Managing Proxy Exclusion List Using REST API](#)

10.7.7.2.1 Managing Proxy Exclusion List Using WLST

Use the WLST commands to add, view, and delete a proxy exclusion list. To manage the proxy exclusion list, you must set the `proxy.exclusion.list` configuration property in the `Agent` category.

Adding a Proxy Exclusion List

To add the proxy exclusion list:

1. Connect to the running instance of the server in the domain for which you want to view the configuration as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Use the `setWSMConfiguration` command and to add the proxy exclusion list.

```
setWSMConfiguration (context,category,name,[group=None],[values=None])
```

Argument	Definition
<code>context</code>	Specify none.
<code>category</code>	Specify Agent.
<code>name</code>	Specify <code>proxy.exclusion.list</code> .
<code>group</code>	Specify none.
<code>values</code>	The array of values. For example, <code>['url1','url2']</code> .

See `setWSMConfiguration` in *WLST Command Reference for Infrastructure Components*.

Example

```
setWSMConfiguration (None,Agent,proxy.exclusion.list,None,['url1','url2'])
```

Viewing the Proxy Exclusion List

Use the `displayWSMConfiguration` command to display the proxy exclusion list.

```
displayWSMConfiguration (context,category,name)
```

See `displayWSMConfiguration` in *WLST Command Reference for Infrastructure Components*.

Example

```
displatWSMConfiguration (None,Agent,proxy.exclusion.list)
```

Deleting the Proxy Exclusion List

Use the `setWSMConfiguration` command and specify `None` for `values` to delete the proxy exclusion list.

```
setWSMConfiguration (context,category,name,[group=None],[values=None])
```

See `setWSMConfiguration` in *WLST Command Reference for Infrastructure Components*.

Example

```
setWSMConfiguration (None,Agent,proxy.exclusion.list,None,None)
```

10.7.7.2.2 Managing Proxy Exclusion List Using REST API

Use the Configuration REST API to add, view, and delete a proxy exclusion list.

Adding a Proxy Exclusion List

To add the proxy exclusion list:

1. Specify the headers on the cURL command line:

```
-H Accept:application/json
```

2. Create a JSON document, `confprop.json`, to update the configuration properties.

The request body contains the details of the update request:

Attribute	Description	Type
<code>name</code>	The name of the property. The property name is system-defined and you cannot edit it.	String
<code>category</code>	The category of the property. The category name is system-defined and you cannot edit it	String
<code>values</code>	The array of values. For example, <code>url1</code> .	Array of string

The following shows an example of the request document. In this example, the category and property names are `Agent` and `proxy.exclusion.list`, respectively; the value for the property is `url1`.

```
[
  {
    "name": "proxy.exclusion.list",
    "category": "Agent",
    "values": [
      "url1"
    ]
  }
]
```

3. Submit a `PUT` request and pass the JSON document defined in the previous step by running the cURL Command:

```
curl -i -X PUT -u username:password -d @file.json -H Content-type:application/json
http://host:port/wsm-pmrest/v2/configuration
```

Example

```
curl -i -X PUT -u Smith:Password -d @reset.json -H Content-type:application/json
http://myhost:7001/wsm-pmrest/v2/configuration
```

The following shows an example of the response indicating the request succeeded.

```
{
  "STATUSCODE": "20032",
  "MESSAGE": "Configuration \"[proxy.exclusion.list of Agent]\" are successfully
updated."
}
```

Viewing the Proxy Exclusion List

Use the `GET` method to display the proxy exclusion list.

```
curl -i -X GET -u username:password -H Content-type:application/json http://host:port/wsm-pmrest/v2/configuration?category={categoryName}&name={propertyName}
```

Example:

```
curl -i -X GET -u Smith:Password -H Content-type:application/json http://myhost:7001/wsm-pmrest/v2/configuration/?category=Agent&name=proxy.exclusion.list
```

The following shows an example of the response body.

```
[
  {
    "name": "proxy.exclusion.list",
    "category": "Agent",
    "source": "/WLS/base_domain",
    "values": [
      "url1"
    ]
  }
]
```

Deleting the Proxy Exclusion List

Use the `PUT` method to delete the proxy exclusion list.

```
curl -i -X PUT -u username:password -H Content-type:application/json http://host:port/wsm-pmrest/v2/configuration?category={categoryName}&name={propertyName}
```

Specify `null` for value to delete the list.

The following shows an example of the response indicating the request succeeded.

```
{
  "STATUSCODE": "20032",
  "MESSAGE": "Configuration \"[proxy.exclusion.list of Agent]\" are successfully updated."
}
```

The following shows an example of the response body.

```
[
  {
    "name": "proxy.exclusion.list",
    "category": "Agent",
    "values": null
  }
]
```

10.7.8 About Support for OAuth 2.0 Tokens with Application Defined Scopes

OWSM supports OAuth2 tokens with application defined scopes. OWSM provides a framework through which a user can provide custom audience condition evaluation logic.

OWSM discovers these conditions using HK2 discovery and checks for its applicability to the current request and if applicable, the conditions are evaluated.

 **Note:**

Custom Implementation Indicates, whether this audience validation is sufficient or necessary or sufficient and necessary. It also evaluates API response to indicate the following:

- If the conditions were evaluated or skipped
- If evaluation result was a success or a failure
- The Response messages.

Contract Details

IAudienceCondition

```
package oracle.wsm.security.audience;

import org.jvnet.hk2.annotations.Contract;

@Contract
public interface IAudienceCondition {
    /**
     * A condition can be of one of three types: SUFFICIENT, NECESSARY or
     SUFFICIENT_AND_NECESSARY. This function
     * is used to determine which type of condition it is
     * @return the type of condition
     */
    public AudienceConditionType getType();

    /**
     * This evaluates the condition to be successful or not, applying the logic specific
     to this condition
     * @param audience the audience string coming from token
     * @param context the data as a Map which is needed for condition to be evaluated
     * @return the condition result containing success or failure result
     */
    public IAudienceConditionResult evaluate(String audience, IAudienceContext context);
}
```

Input Interface

IAudienceContext

```
package oracle.wsm.security.audience;

public interface IAudienceContext {
    /**
     * This method is used for retrieving the message context used for this request
     * @return the MessageContext
     */
    public IMessageContext getMessageContext();

    /**
     * This method is used for retrieving the claims in the incoming JWT token
     * @return the Jwt Claims in the form of Map
     */
    public Map<String, Object> getJwtClaims();
}
```

```

/**
 * This method is used for retrieving HTTP header from the transport
 * @return the HTTP headers
 */
public Map<String, String> getHTTPHeaders();
/**
 * This method is used for retrieving the value of property from the Map
 * @param the property name
 * @return the property value
 */
public Object getProperty(String propName);
}

```

Response Interface

IAudienceConditionResult

```

package oracle.wsm.security.audience;

public interface IAudienceConditionResult {
/**
 * This returns condition status if it was successful or not when the condition was
evaluated
 * @return true if condition is successfully enforced
 */
public boolean isEvaluationSuccessful();
/**
 * Used by OWSM engine to identify scenarios where condition is not enforced or was
not meant to be enforced but evaluate is invoked
 * @return true if condition is present and enforced, false if it is skipped
 */
public boolean isConditionPresent();

/**
 * The condition maintains list of messages (error or non error) when condition is
evaluated,
 * which need to be logged by the caller of this method
 * @return list of response messages to be logged by OWSM engine. It can be an error
message.
 */
public List<String> getResposneMessages();
}

```

10.8 Understanding OWSM integration with 3rd Party Servers

OWSM is integrated with 3rd party servers like Google OAuth2 server and Twitter OAuth server.

This section describes how to enable OWSM integration.

- [OWSM Integration with Twitter OAuth server](#)
- [OWSM Integration with Google OAuth2 server](#)

10.8.1 About OWSM Integration with Twitter OAuth server

OWSM is integrated with twitter. Twitter uses OAuth1 to provide authorized access to its API .

The REST APIs provides programmatic access to read and write Twitter data. Author a new Tweet, read author profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth. Responses are available in JSON.

OWSM supports single-user OAuth use case, which requires providing partial support of [OAuth 1.0](#) protocol in OWSM to use OAuth1 type of access token and token secret to secure the request to Twitter API. In this case, we pick up from the point where we are working with an access token to make signed requests for Twitter resources.

OWSM provides new OAuth1 client policy which allows applications to use Twitter API using the statically generated consumer and access tokens. Retrieval of access token is not done by OWSM policy.

To integrate Twitter account with OWSM do the following:

- Prerequisites for configuring Twitter account
- Configuring Twitter account for integration with OWSM

10.8.1.1 Prerequisites for configuring Twitter account

Twitter currently uses OAuth1 protocol for most of the authentication methods. OAuth2 bearer token is only supported for application-only authentication. We will focus on single-user OAuth use case, which requires providing partial support of OAuth1 protocol in OWSM to use OAuth1 type of access token and token secret to secure the request to Twitter API.

To use OAuth, an application must:

- [Obtain access tokens](#) to act on behalf of a user account.
- [Authorize all HTTP requests](#) it sends to Twitter's APIs using access token.

10.8.1.2 Configuring Twitter account for integration with OWSM

After you obtain the access tokens, you can configure the twitter account. Follow the steps to create and run a rest application which tweets a message (updates status):

Twitter Account setup

1. Create Twitter user account at <https://twitter.com/>.
2. Using this account, go to <https://apps.twitter.com>.
3. Click **Create New App** to create new application. Fill out all details.
4. Click on the application.
5. Click on tab **Key and Access Tokens**.
6. Generate Consumer Key and Secret, Access token and Secret.
7. Note down all four keys as you will need them later - Consumer Key, Consumer Secret, Access Token and access Token Secret.

Configure credentials

1. Create a CSF key which holds the value of consumer key and consumer secret.

For example: `updateCred(map="oracle.wsm.security",
key="basic.client.oauth1.credentials", user="NMugHF4zz3ywchygNsXHy",
password="XhcrhgVzAHWmerW4JN20xLrv3FliwlbIYPqTnbntUyY9wB", desc="Twitter
account consumer key")`

2. Create a CSF key which holds the value of access token and access token secret.

For example: `updateCred(map="oracle.wsm.security",
key="basic.token.oauth1.credentials",`


```
user="778014527828787200-3e7LqGPB79ve0Bbo8Z0acv17op2",  
password="ZAVO3BsBNxGMSr7kwGeX3efhqQ2psQvcdvuLLCYXb", desc="Twitter account  
access token")
```

Configure the truststore

The Twitter SSL certificate needs to be imported into your keystore

1. Export the twitter SSL certificate into a local file. Let's call it twitter.crt. You can export the SSL certificate using any browser. For example, if you are using Firefox, you can export as follows :
 - Go to <https://api.twitter.com/1.1/statuses/update.json>.
 - Click on the SSL certificate icon at the top / Padlock at the bottom.
 - Click on the **Details** Tab
 - Chose which certificate you want from the hierarchy [not circled in picture]
 - Click **Export**
2. Import this certificate into your truststore. For example, if you are using JKS, you will import it as follows:
 - `keytool -import -keystore client-truststore.jks -storepass changeit -file twitter.crt`

Create a RESTful Client using Jdeveloper

In order to test the integration with Twitter OAuth server, we need to create a web application that will invoke one of the Twitter APIs using the OWSM OAuth1 Twitter Client policy. In the sample provided here we are trying to access the Tweet API at <https://api.twitter.com/1.1/statuses/update.json> URL.

To create the client application do the following:

1. Start JDeveloper.
2. Create a new Application and Project.
3. Right Click on the new Project created above and select New HTTP Servlet. (if it does not show up there, from the New from Gallery menu, select Web Tier, Servlets, and then HTTP Servlet).
4. Edit this new servlet with the content provided in the code block below.
 - [Twitter REST Client sample code for POST request](#)
 - [Twitter REST Client sample code for GET request](#)
5. Add required OWSM and JERSEY libraries as needed.
6. Save the Project and confirm that all imports have been resolved and there are no build issues.
7. Deploy your web application to the WLS server. Right Click on the project and select Deploy and then webapp. Deploy directly to the Application Server if you have one configured in JDEV, else deploy to a WAR file and manually deploy this WAR file via the WLS console.

Test the webapp from the browser. You should receive Status 200 from the Twitter API along with some other metadata printed on the browser. You can login to your account www.twitter.com and observe the tweet you sent using OWSM policy.

10.8.2 About OWSM Integration with Google OAUTH2 Server

OWSM OAuth2 clients can integrate with Google OAUTH2 server using the Google OAuth2 Client SSL Policy. Google APIs use the OAuth 2.0 protocol for authenticating and authorizing the client application that requires access.

For OWSM, Google OAuth 2.0 supports server-to-server interactions between a server application and Google APIs. This is a 2-legged OAuth2 scenario, where the server application obtains an access token (AT) from the OAuth server by sending a JWT token and then uses the AT to access the Google APIs. For this scenario you need a service account, which is an account that belongs to your application instead of to an individual end user. The client application invokes the Google APIs on behalf of the service account and not the user.

To integrate OWSM OAuth 2.0 client with Google OAUTH2 server, do the following:

- [Creating a Google Project Using the Google API Console](#)
- [Configuring Google Service Account](#)
- [Enabling Google API](#)
- [Configuring the Keystore for the Client Application](#)
- [Verifying Integration with Google OAuth Endpoints Using JDeveloper](#)

10.8.2.1 Creating a Google Project Using the Google API Console

Each client will have unique credentials to access Google OAUTH2 Server. You need to register the client application with the Google API Console to obtain the client credentials such as a client ID and client secret that are known to both Google and your application. Create the Google API Console project and client ID, as described in Google Sign-In for Websites documentation <https://developers.google.com/identity/sign-in/web/devconsole-project>.

10.8.2.2 Configuring Google Service Account

You must configure a Google Service account to integrate OWSM client with Google OAUTH2 server.

Create a Google Service Account

You must create a Google service account, as described in the Google Identity Platform documentation <https://developers.google.com/identity/protocols/OAuth2ServiceAccount#creatinganaccount>

Note:

1. Keep a note of the Private/Public key generated by Google on your machine as it cannot be re-generated if lost.
2. Keep a note of the password for the private key as it will not be displayed again.

Access Token Request and Response

Client application request Access Token from the Google Authorization server to access Google APIs.

A single access token can grant varying degrees of access to multiple APIs. A variable parameter called scope controls the set of resources and operations that an access token permits. During the access-token request, your application sends one or more values in the scope parameter.

Permission to perform this type of impersonation must be granted before an application can impersonate a user, and is usually handled by a domain administrator. For more information on domain administration, see [Managing API client access](#).

Upon successful verification of client's request by the OAUTH2 server, client extracts the access token from the response obtained and sends the token to the Google API that it needs to access. Access tokens are valid only for the set of operations and resources described in the scope of the token request.



Note:

Handling Access Token Expiration: The access token once obtained will be cached until it expires. When it expires OWSM will automatically obtain another access token and cache it until it expires.

10.8.2.3 Enabling Google API

You must enable the API(s) on the Google Developers Console, under the project created so that the Google OAuth2 server is able to validate the client's request for an Access Token for any given Google API.

Follow the steps below to enable the API(s) which your application needs to access:

1. Login to the Google Developers Console using the required credentials.
2. Click on Overview in the Google API Manager page.
3. Check the Enabled APIs tab and see if the API you want to access is already enabled. If it is not enabled, go to Google APIs tab and search the API you are looking for.
4. Click on the required API and then click the Enable button on the top.
5. Go back to the Enabled APIs tab and verify that the newly enabled API shows up in the list.

10.8.2.4 Configuring the Keystore for the Client Application

Before you configure the keystore for the client application, ensure that you have created the Java keystore (JKS) for the OWSM domain. See [Generating Private Keys and Creating the Java Keystore](#).

Follow the steps below to configure the keystore:

1. Use the `keytool` utility to import the PKCS#12 keystore (This keystore is obtained after Creating the Google Service Account) into the JKS keystore.

```
keytool -importkeystore -srckeystore <location of the .p12 file>-srcstoretype PKCS12  
-destkeystore <OWSM keystore location> -deststoretype JKS
```

Enter the src and dest keystore pwds. (Hint: The src password is the one noted down after creating the Google Service Account and verify that an entry has been added to the keystore , "privatekey".

2. Use the `keytool -list` command to view the contents of the keystore:

```
keytool -keystore <OWSM keystore location> -list
```

3. Use WLST to import one or more aliases from the keystore using the `importKeyStore` script at the command line:

```
[userx@hostname bin]$ ./wlst.sh
```

```
Initializing WebLogic Scripting Tool (WLST) ...
```

```
Welcome to WebLogic Server Administration Scripting Shell
```

```
Type help() for help on available commands
```

```
wls:/offline> connect("weblogic","password","t3://myhost.example.com:8001")
Connecting to t3://myhost.example.com:8001 with userid weblogic ...
Successfully connected to Admin Server "jrfServer_admin" that belongs to domain
"jrfServer_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:/jrfServer_domain/serverConfig/> svc = getOpssService(name='KeyStoreService')
```

```
wls:/jrfServer_domain/serverConfig/> svc.importKeyStore(appStripe='owsm',
name='keystore', password='password', aliases='privatekey',
keypasswords='notasecret', type='JKS', permission='true', filepath='<OWSM keystore
location>')
```

4. Create a new credential, as per the steps mentioned [here](#), say `google-service-credential`, in the `cwallet.sso`. The name of the credential will be the service account email ID which is generated after creating the Google Service Account. Password does not matter.

10.8.2.5 Verifying Integration with Google OAuth Endpoints Using JDeveloper

In order to test the integration with Google OAuth2 server, you need to create a web application that will invoke one of the Google APIs using the OWSM OAuth2 Google Client policy to obtain the Access token. In the sample provided here we are trying to access the Google Drive API at :

```
"https://www.googleapis.com/drive/v2/files" URL.
```

Follow the steps to create the client application:

1. Starting Oracle JDeveloper:

- a. UNIX Systems:

- Navigate to the following location on your system: `JDEV_HOME/jdeveloper/jdev/bin/`
- Run the following command: `./jdev`

- b. Windows:

Start Oracle JDeveloper from the command line by running one of the following commands:

- `JDEV_HOME\jdeveloper\jdeveloper.exe`
- `JDEV_HOME\jdeveloper\jdev\bin\jdevw.exe`

2. Create a Custom Application using JDeveloper, as described in [How to Create a Custom Application in Oracle Fusion Middleware Developing Applications with Oracle JDeveloper](#).
3. Delete the default project that is auto-created for the application.

4. Creating a New Custom Project:
 - a. In the Applications window, open the application that will contain the new project.
 - b. Click the Application Menu icon, and select New Project to open the Projects page of the New Gallery.
 - c. Under Items, select Custom Project.
 - d. Click OK.
5. Generate the HTTP Servlet:
 - a. In the Applications window, select the project in which you want to create the new servlet.
 - b. From the main menu, choose File > New > From Gallery > Web Tier > Servlets, or right-click and choose New. The New Gallery opens.
 - c. In the Items list, double-click HTTP Servlet to launch the Create HTTP Servlet wizard.
6. Edit this new HTTP servlet with the content provided in the code block below:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;
import org.glassfish.jersey.client.ClientConfig;
import oracle.wsm.metadata.feature.AbstractPolicyFeature;
import oracle.wsm.metadata.feature.PolicyReferenceFeature;
import oracle.wsm.metadata.feature.PolicySetFeature;
import oracle.wsm.metadata.feature.PropertyFeature;
import oracle.wsm.security.util.SecurityConstants;
@WebServlet(name = "TestClient", urlPatterns = { "/testclient" })
public class TestClient extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=UTF-8";
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title><TestClient>/title</title></head>");
        out.println("<body>");
        out.println("<p>The servlet has received a GET. This is the reply.</p>");
        Client client = null;
        ClientConfig cc = null;
        try {
            String BASE_URI = "https://www.googleapis.com/drive/v2/files";
            PropertyFeature tokenUri = new PropertyFeature(
                SecurityConstants.ConfigOverride.CO_TOKEN_URI,
                "https://accounts.google.com/o/oauth2/token");
            PropertyFeature csfKey = new PropertyFeature(
                SecurityConstants.ConfigOverride.CO_CSF_KEY,
                "google-service-credential");
            PropertyFeature oauth2CsfKey = new PropertyFeature(
                SecurityConstants.ConfigOverride.CO_OAUTH2_CLIENT_CSF_KEY,
                "google-service-credential");
            PropertyFeature signCsfKey = new PropertyFeature(
```

```

SecurityConstants.ConfigOverride.CO_SIG_CSF_KEY,
"privatekey");
PropertyFeature scope = new PropertyFeature(
    SecurityConstants.ConfigOverride.CO_CUSTOM_JWT_CLAIMS,
    "scope=https://www.googleapis.com/auth/drive");
PolicyReferenceFeature[] clientPRF = new
PolicyReferenceFeature[] {
    new PolicyReferenceFeature(
        "oracle/oauth2_config_client_policy", tokenUri),
    new PolicyReferenceFeature(
        "oracle/
http_oauth2_token_over_ssl_google_jwt_client_policy",
        new PropertyFeature[] { csfKey, oauth2CsfKey,
signCsfKey, scope } ) };
cc = new ClientConfig();
cc.property(AbstractPolicyFeature.ABSTRACT_POLICY_FEATURE,
    new PolicySetFeature(clientPRF));
client = ClientBuilder.newClient(cc);
WebTarget webTarget = client.target(BASE_URI);
Response rest_response = webTarget.request("application/
json").get(
    Response.class);
int status = rest_response.getStatus();
String textEntity = rest_response.readEntity(String.class);
out.println("Response status :" + status);
    out.println("Response from Google Drive API :" + textEntity);
    out.println("</body></html>");
} finally {
    client.close();
    out.close();
}
}
}
}

```

7. Add the JAX-RS Jersey 2.x library to the project, as follows:
 - a. With the project selected in the Applications window, open the Project Properties dialog.
 - b. Select the **Libraries and Classpath** node.
 - c. Click Add **Library**.
 - d. Select **JAX - RS Jersey 2.x** under Extension and click OK.
8. Add the wsm-policy-core.jar and wsm-secpol.jar files to the project, as follows:
 - a. With the project selected in the Applications window, open the Project Properties dialog.
 - b. Select the **Libraries and Classpath** node.
 - c. On the Libraries and Classpath page, click **Add JAR/Directory**.
 - d. Click browse and to navigate your JDEV home or JDEV install location/
oracle_common/module/oracle.wsm.common/wsm - policy - core.jar and Open.
 - e. Repeat the above step to install the wsm-secpol.jar file.
9. Save the Project and confirm that all imports have been resolved and there are no build issues.
10. Deploy the web application using Oracle WebLogic Server, as described in Deploying Applications in Oracle Fusion Middleware Developing Applications with Oracle JDeveloper.

11. You must add the `WSIdentityPermission` to the client application (For example, `webapp`), this allows the client application to act on behalf of the identity presented in the `csf-key`. Add the permission by running the `WLST` command, as follows:

```
grantPermission(appStripe=None, codeBaseURL='file:${common.components.home}/modules/  
oracle.wsm.agent.common_${jrf.version}/wsm-agent-  
core.jar',principalClass=None,principalName=None,permClass='oracle.wsm.security.WSIde  
ntityPermission',permTarget='resource=webapp',permActions='assert')
```

12. Test the application by running the following URL on the browser `http://hostname:port/GoogleOAuthClient/testclient`. Verify that you should receive Status 200 from the Google Drive API along with some other metadata printed on the browser.

11

Configuring Authentication Using Oracle Web Services Manager

OWSM provides security policies that provide authentication only, and authentication with message protection.

For a summary of the authentication policies available in the current release, see "[Authentication Only Policies](#)" and "[Message Protection and Authentication Policies](#)" in [Determining Which Predefined Policies to Use for a Web Service](#). For more information on authentication, see "Understanding Authentication" in *Understanding Oracle Web Services Manager*.

This chapter includes the following sections:

- [Overview of Authentication Configuration](#)
- [Supported Authentication Providers in WebLogic Server](#)
- [About Configuring Digest Authentication](#)
- [About SAML Configuration](#)
- [About Propagating Identity Context with OWSM](#)
- [Understanding Kerberos Token Configuration](#)
- [About WS-Trust Configuration](#)

11.1 Overview of Authentication Configuration

Authentication means verifying that the user is who one claim to be. A user's identity is verified based on the credentials presented by that user, such as username/password, digital certificate, standard Security Assertion Markup Language (SAML) token, or Kerberos token.

Both SAML and Kerberos policies require some configuration on both the service and the client side. In addition to policies, SAML and Kerberos have associated login modules. These modules, which are typically configured through Fusion Middleware Control, control access to the web service.

In SAML tokens, user attributes, user roles, and issuer name can be added to verify against the policy. These attributes are verified by the Oracle Platform Security Services (OPSS). For more information, see [About SAML Configuration](#).

In Kerberos, you initialize and configure a Key Distribution Center (KDC) for use by the web service client and web service. You create principals in the KDC user registry and configure the service and the client to use the correct KDC. [Understanding Kerberos Token Configuration](#).

For a list of all of the policies that pertain to authentication, see [Authentication Only Policies](#), [Message Protection and Authentication Policies](#), and [WS-Trust Policies](#).

11.2 Supported Authentication Providers in WebLogic Server

Policies that use any of the supported token types—including username, X.509, Kerberos, SAML, HTTP BASIC, and so forth—require you to configure an authentication provider, such as the WebLogic Default Authentication provider, in WebLogic Server.

For WebLogic Server configuration details, see "Configuring Authentication Providers" in *Administering Security for Oracle WebLogic Server 14c (14.1.2)*.

For a list of the authentication policies available in the current release, see "[Authentication Only Policies](#)" and "[Message Protection and Authentication Policies](#)" in [Determining Which Predefined Policies to Use for a Web Service](#).

For details about WebLogic Server security, see the following documents:

- *Administering Security for Oracle WebLogic Server 14c (14.1.2)*
- Oracle WebLogic Server Administration Console Online Help

You can configure any of the following WebLogic Server Authentication providers for use with OWSM, regardless of the token type used in the OWSM policy:

- 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 OWSM generates. This limitation does not exist for any of the LDAP-based providers.

With OWSM, you do not need to configure a WebLogic Server provider for specific token types, such as SAML, Kerberos, and X.509 tokens. Specifically, the OWSM runtime does *not* use any other WebLogic Server providers, including but not limited to:

- X509 providers—OWSM policies based on an X509 token do not use the WebLogic Server X509 Identity Assertion provider.
- SAML providers—OWSM 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

11.3 About Configuring Digest Authentication

Digest authentication is an authentication mechanism in which a Web application authenticates itself to a web service by sending the server a digest, which is a cryptographic hash of the password, nonce, and timestamp.

OWSM supports digest based authentication for all of the username-token authentication policies.

This section includes the following topics:

- [Prerequisites for Configuring Digest Authentication](#)
- [Configuring the Default Authenticator and Identity Asserter](#)
- [Attaching a Policy and Enabling Digest Authentication](#)

11.3.1 Prerequisites for Configuring Digest Authentication

You have to fulfill certain prerequisites for configuring Digest Authentication.

The host system on which the web service is deployed must be a WebLogic Server system configured with the embedded LDAP authentication provider (Default Authenticator). This is because the web service client sends a password in the form of a digest and, in order to authenticate it at the service side, the authentication provider must recreate the digest using the clear-text password, the nonce, and timestamp. The Default Authenticator is able to get the password in the required clear text.

There is no limitation for the digest-based client policies. The web service client to which they are attached can be run from all supported platforms.

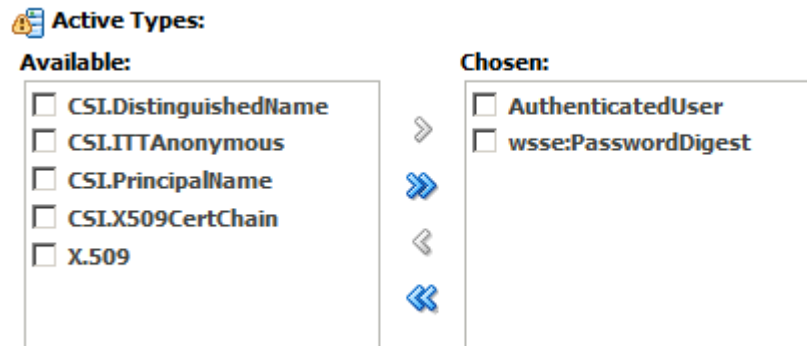
11.3.2 Configuring the Default Authenticator and Identity Asserter

You can configure the Default Authenticator and Default Identity Asserter from the Weblogic Server Remote Console.

Perform the following steps from the WebLogic Server Remote Console to configure the Default Authenticator and Default Identity Asserter for the security realm:

1. Set the Enable Password Digests checkbox on the provider-specific page of the Default Authenticator. See [Configure Authentication and Identity Assertion providers](#) for more information.
2. Set `wsse:PasswordDigest` as an active type on the common page of the `DefaultIdentityAsserter`, as shown in [Figure 11-1](#). See [Configure Authentication and Identity Assertion providers](#) for more information.

Figure 11-1 wsse:PasswordDigest as Active Type



3. Restart WebLogic Server.
4. Users created after you configure digest authentication can be authenticated using the digest based mechanism.

However, users created before you configure digest authentication cannot be authenticated using the digest based mechanism because the WebLogic Server security runtime has already stored the password digests (rather than cleartext passwords) for the already-created users.

You must recreate these users so that their cleartext password, and not their digest, is stored in the password database.

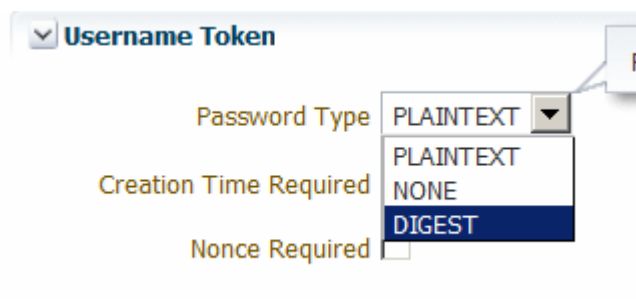
11.3.3 Attaching a Policy and Enabling Digest Authentication

This section describes how to attach a policy and in turn enable digest authentication.

To attach a policy and in turn enable digest authentication, perform the following steps:

1. If you have not already done so, attach a copy of one of the web service policies that support digest authentication. See [Attaching Policies to Manage and Secure Web Services](#) for information about how to copy and attach a policy.
2. Open the policy and change the **Password Type** drop-down menu to Digest, as shown in [Figure 11-2](#).

Figure 11-2 Selecting Digest From the Password Type Drop-Down Menu



3. Set the **Creation Time Required** control.
4. Set the **Nonce Required** control.
5. Repeat steps one through four for the client version of the policy.

**Note:**

You must select Digest as the password type and set both the **Creation Time Required** and **Nonce Required** controls for the policy to be valid.

11.4 About SAML Configuration

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.

The predefined SAML policies include `saml` in their name and are listed in [Oracle Web Services Manager Predefined Policies](#).

**Note:**

You cannot edit the predefined SAML policies. To modify any of the settings or configuration properties, you must either clone the policy to edit it, or set the configuration properties using a configuration override after attaching the policy.

The following sections provide more information about SAML configuration:

- [Overview of Flow of SAML Token Validation](#)
- [Configuring SAML Web Service Client at Design Time](#)
- [Including User Attributes in the Assertion](#)
- [Including User Roles in the Assertion](#)
- [Understanding the Configuration of Oracle Platform Security Services \(OPSS\) for SAML Policies](#)
- [Adding an Additional SAML Assertion Issuer Name](#)
- [About SAML Web Service Client Configuration for Identity Switching](#)
- [Understanding Trusted Issuers and Trusted Distinguished Names List for SAML Signing Certificates](#)
- [Understanding How to Use Anonymous Users with SAML Policies](#)

11.4.1 Overview of Flow of SAML Token Validation

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.

Any configured Authentication provider as described in [Supported Authentication Providers in WebLogic Server](#) can then be invoked.

For more information, refer to the following sections:

- [Validating a SAML Assertion](#)
- [Use Cases for SAML Token Validation](#)

11.4.1.1 Validating a SAML Assertion

This section describes how to validate a SAML assertion.

Depending on the types of SAML assertions and the mechanisms used to protect the messages, one or more of the following steps are used to validate a SAML assertion:

1. Verify the signature on the SAML assertion:
 - a. Obtain the signing certificate included or referenced in the signature for the SAML assertion.
 - b. Verify the certificate is trusted against the certificate chain present in the key store.
 - c. Verify the SAML assertion signature with the public key in the signing certificate.
2. Validate the SAML assertion conditions:
 - Not before and not after.
 - SAML audience restriction.
3. Validate the trust for the SAML assertion issuer:
 - a. Validate if the issuer is trusted.
 - b. Validate if the DN of the signing certificate is trusted.
 - c. Token attribute rule.
4. Authenticate the user in the SAML assertion.

Note:

Typically, the entire message is signed and encrypted. Consider for example the [oracle/wss10_saml_token_with_message_protection_client_policy](#). By default, when using this policy the entire message is signed and encrypted. However, as described in [About Configuring Partial Encryption with Fusion Middleware Control](#), the assertion templates support partial signing and encryption as well as full signing and encryption of the message body.

If you do not sign or encrypt any part of the message body, then only the SAML token is signed.

In the previous example of the [oracle/wss10_saml_token_with_message_protection_client_policy](#) policy, for SAML sender vouches the SAML token is signed by the client's private certificate.

11.4.1.2 Use Cases for SAML Token Validation

This section describes how SAML tokens are validated in various use cases.

[Table 11-1](#) lists combinations of different SAML types with different security mechanisms and how a SAML assertion is validated in each case.

Table 11-1 SAML Token Validation

Use Case	Description	How Validated
SAML sender vouches with message protection.	SAML sender vouches over message protection has signing, but in this case the message body and the SAML token are signed together in one signature with the client's private certificate.	Steps: 1-4
SAML sender vouches with 2-way-SSL. For example, oracle/wss_saml20_token_over_ssl_service_policy	Message protection is via SSL only. The SAML token is signed by the client's private certificate.	Steps: 1b. Validate that the client SSL certificate is trusted. 2, 3, 4
SAML sender vouches with no security. For example, oracle/wss10_saml20_token_service_policy	This policy is not secure and is provided for demonstration purposes only. Although the SAML issuer name is present, the SAML token is not signed. The credentials in the SAML 2.0 token are authenticated against a SAML 2.0 login module.	Steps: 2, 3a, 4
SAML token bearer with 1-way-SSL. For example, oracle/wss_saml20_token_bearer_over_ssl_service_policy	In the case of SAML bearer, the SAML tokens are signed and the signature is present inside the SAML token. This is independent of whether the body is signed.	Steps: 1, 2, 3, 4
SAML unsigned token bearer with 1-way-SSL. (Disabled by default, but can be turned on for backward compatibility.)	The SAML token is not signed.	Steps: 2, 3, 4
SAML holder of key with message protection. The entire body is signed with the HOK.	The SAML tokens are signed and the signature is present inside the SAML token. The body is also signed with the HOK.	Steps: 1, 2, 3, 4 Additional step 5. Validate that the sender has the secret key by verifying the signature with the key.
SAML holder of key with timestamp signing only. The message body is not signed.	The SAML tokens are signed and the signature is present inside the SAML token. The timestamp is also signed with the HOK.	Steps: 1, 2,3,4 Additional step 5. Validate that the sender has the secret key by verifying the signature with the key.

11.4.2 Configuring SAML Web Service Client at Design Time

You have to configure the username for SAML assertion.

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.

To configure the username for SAML assertion:

For a JSE client application, configure the username as a `BindingProvider` property as follows:

```
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.

11.4.3 Including User Attributes in the Assertion

You can use configuration property to add user attributes to the SAML assertion.

SAML client policies include the `user.attributes` configuration 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 that the `subject.precedence` configuration property is set to `true`. (If `subject.precedence` is `true`, the user name to create the SAML assertion is obtained only from the Subject.)

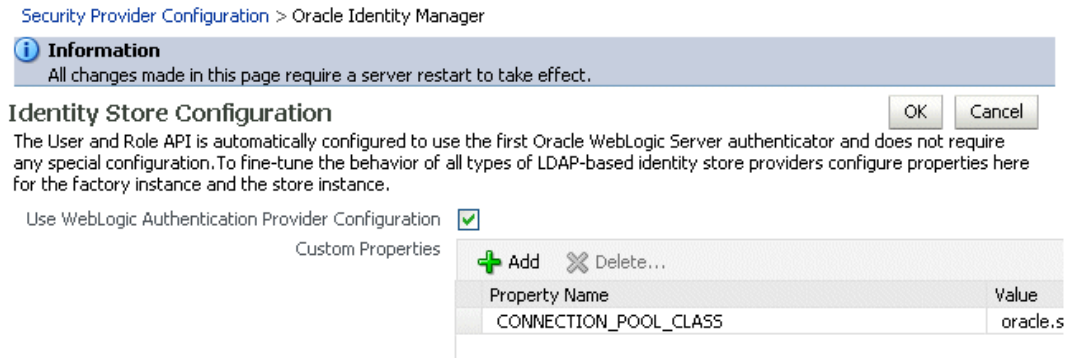
The OWSM runtime reads the values for these attributes from the configured identity store, and then includes the attributes and their values in the SAML assertion.

The `user.attributes` configuration 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 [Supported Authentication Providers in WebLogic Server](#).

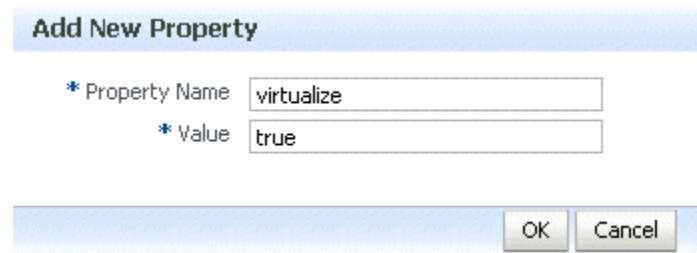
If you have more than one identity store configured, and you want to search for the user in all identity stores, follow these steps to enable searching in all configured identity stores.

1. In the Target Navigation pane, expand **WebLogic Domain** to show the domain for which you need to configure the identity store provider. Select the domain.
2. From the **WebLogic Domain** menu, select **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 11-3](#).

Figure 11-3 Identity Store Configuration Page

- Click **Add** to add a custom property.
The Add New Property window is displayed.
- Add the property "virtualize" with a value of "true" as shown in [Figure 11-4](#).

Figure 11-4 Adding the virtualize property

- Click **OK** to submit the changes.
- Click **OK** on the Identity Store Configuration page.
- Restart the server.

11.4.4 Including User Roles in the Assertion

You can pass the user's role as an attribute statement in the SAML assertion

To pass the user's role as an attribute statement in the SAML assertion, 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.

11.4.5 Understanding the Configuration of Oracle Platform Security Services (OPSS) for SAML Policies

You have to configure OPSS for the predefined SAML policies.

To configure OPSS for the predefined SAML policies, follow these steps:

- Configure the SAML login module, as described in [Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#).

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](#) for information on adding an additional issuer.

2. Configure the Authentication provider in the WebLogic Server Remote 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 [Overview of Configuring Keystores for Message Protection](#).
4. If you will be using policies that require SSL, you need to configure SSL, as described in [About Configuring Keystores for SSL](#).

11.4.6 Adding an Additional SAML Assertion Issuer Name

There are three scenarios in which you need to add additional issuers to the trusted issuers list.

If you are using the predefined SAML policies (or assertions) on both the web service client and web service sides, the SAML issuer name is specified as `www.oracle.com`. Therefore, you can generally use the defaults and not configure any issuer.

- You specified a trusted issuer using the `saml.trusted.issuers` configuration override property in a SAML predefined web service policy or assertion. Trusted issuers defined here will apply at the application level and will override trusted issuers defined at the domain level.
- You specified a SAML issuer using the `saml.issuer.name` configuration property in a SAML predefined web service client policy or assertion.
- If a different client, for instance .NET/STS, is talking to a web service protected by a predefined SAML policy.

The preferred method for adding SAML issuers is to use domain-level configuration, as described in the following sections:

- [SAML Trusted Issuers and DN Lists Using Fusion Middleware Control](#).
- [LINK TO WLST Section](#)

Note that when you add an issuer as described above, it applies to all services in the domain and a server restart is not required.

For backward compatibility, you can also add an issuer in Fusion Middleware Control by configuring the SAML or SAML2 login module in the security provider, although the preferred method is to use the domain level configuration. If you do define the SAML issuers in the SAML login module, the issuers are persisted in `jps-config.xml`, and you must restart the servers in the domain for the changes to take effect.

 **Note:**

OWSM uses the following hierarchy to determine the trusted issuers to be used at runtime:

- First, the list of trusted issuers configured at the policy level is checked and used. This will override trusted issuers at the domain level.
- If no trusted issuers are defined at the policy level, the domain level configuration is checked and used.
- Lastly, if no trusted issuers are defined at either the policy or domain level, then the issuers list defined in the SAML login module is checked and used.

To add an additional SAML assertion issuer to the Issuers list in the SAML login module:

1. In the Target Navigation pane, expand **WebLogic Domain** to show the domain for which you need to add the issuer. Select the domain.
2. In the content pane, select **WebLogic Domain**, then **Security**, and then **Security Provider Configuration**.
3. In the Login Modules section, select the SAML or SAML2 login module as appropriate and click **Edit**.

The Edit Login Module page is displayed.

4. In the SAML Specific Attributes section of the page, click **Add**.
5. In the Add Issuer window, enter the issuer name in the **Issuer** field and click **OK**.
6. Restart the server.
7. Configure the issuer in the client policy as follows:
 - Post-deployment, specify a configuration override value for the `saml.issuer.name` property in an attached SAML client policy. If you cloned a predefined Oracle SAML client policy, you can set the value in the `saml.issuer.name` configuration property before attaching the policy.
 - At design time, set the `saml.issuer.name` property in the `BindingProvider`.

11.4.7 About SAML Web Service Client Configuration for Identity Switching

Identity switching means that the policy propagates a different identity than the one based on the authenticated Subject.

OWSM includes the `wss11_saml_token_identity_switch_with_message_protection_client_policy` policy that enables identity switching. Identity switching means that the policy propagates a different identity than the one based on the authenticated Subject. To explore more about identity switching, view the following topics:

- [Understanding Identity Switching Use Case Scenarios](#)
- [Setting the `javax.xml.ws.security.auth.username` Property](#)
- [Setting the Permission Using `WSIdentityPermission`](#)

11.4.7.1 Understanding Identity Switching Use Case Scenarios

You might have a scenario in which your SOA 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 token in the outbound web service request. 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.

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`.

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` policy. It communicates with the SOA entry point as user `end_user1`.
- The SOA entry point is protected by `wss11_username_with_message_protection_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 the `wss11_saml_identity_switch_message_protection_client_policy` policy 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. 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 to add the `WSIdentityPermission` permission to the SOA reference binding component.

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 OWSM accepts the externally-supplied identity must have the `WSIdentityPermission` permission.

This is to avoid potentially rogue applications from providing an identity to OWSM.

 **Note:**

The `wss11_saml_token_identity_switch_with_message_protection_client_policy` policy disables local optimization (see "[Using Local Optimization with OWSM Policies \(SOA Composites\)](#)" for SOA to SOA interactions on the same server.)

This policy is compatible with the `wss11_saml_token_with_message_protection_service_policy` policy on the web service.

11.4.7.2 Setting the `javax.xml.ws.security.auth.username` Property

This section gives you information about SOA and BPEL process for setting username property.

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.

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.

11.4.7.3 Setting the Permission Using `WSIdentityPermission`

This procedure shows how to set permission to the SOA reference binding component as a System Grant.

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.

To use Fusion Middleware Control to add the `oracle.wsm.security.WSIdentityPermission` permission to the SOA reference binding component as a System Grant, perform the following steps:

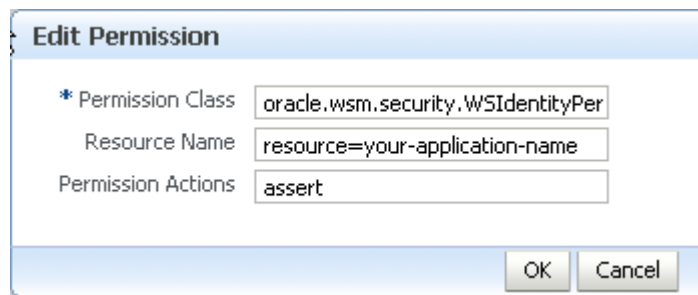
1. In the Target Navigation pane, expand **WebLogic Domain** to show the domain for which you need to configure the application. Select the domain.
2. Using Fusion Middleware Control, click **WebLogic Domain**, then **Security**, and then **System Policies**. System policies are the system-wide policies 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:${common.components.home}/modules/oracle.wsm.common_${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 11-5](#).

Figure 11-5 Editing the `WSIdentityPermission`



To use WLST to add the `oracle.wsm.security.WSIdentityPermission` permission, execute the following command:

```
grantPermission (codeBaseURL="file:${common.components.home}/modules/
oracle.wsm.common_${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"`.

11.4.8 Understanding Trusted Issuers and Trusted Distinguished Names List for SAML Signing Certificates

You can define a list of trusted distinguish names (DNs) for SAML signing certificates for additional security.

By default, OWSM checks the incoming issuer name against the list of configured issuers, and checks the SAML signature against the configured certificates in the OWSM keystore. If you define a trusted DN list, OWSM also verifies that the SAML signature is signed by the particular certificate(s) that is associated with that issuer.

Configuration of the trusted DN 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 OWSM allows signing by any certificate, as long as that certificate is trusted by the certificates present in the OWSM keystore.

For more information about defining a trusted DNs list for SAML signing certificates, see [SAML Trusted Issuers and DN Lists Using Fusion Middleware Control](#).

11.4.9 Understanding How to Use Anonymous Users with SAML Policies

Allowing anonymous users over SAML is provided as a convenience so that you can have one policy that supports both authenticated and anonymous users.

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 OWSM.

11.5 About Propagating Identity Context with OWSM

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.

This section contains the following topics:

- [Overview of Identity Context](#)
- [Propagating Identity Context Using SAML Policies](#)
- [Configuring Identity Context Propagation: Main Steps](#)

11.5.1 Overview of Identity Context

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.

OWSM 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 Java EE (WebLogic) 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*.

11.5.2 Propagating Identity Context Using SAML Policies

OWSM propagates the Identity Context from web service clients to web services via SAML 1.1 or SAML 2.0 assertions.

To use this feature, you must specifically enable identity context propagation by using the `propagate.identity.context` configuration property for both the web service client policy and the web service policy. That is, OWSM can propagate the Identity Context only if you specifically allow both the web service client policy and web service policy to do so.

OWSM propagates the Identity Context from web service clients to web services via SAML 1.1 or SAML 2.0 assertions. Therefore, only SAML policies include the `propagate.identity.context` configuration property.

The OWSM policies that support identity context are listed in [OWSM Policies Supported for Identity Context](#).

11.5.3 Configuring Identity Context Propagation: Main Steps

You can specify a value on the Configurations page in the policy or override it when you attach the policy.

You can specify a value for `propagate.identity.context` on the Configurations page in the policy as described in this section, or override it when you attach the policy.

 **Note:**

If you choose to set this property in the policy, you need to make copies (both client and server) of the desired predefined SAML policies listed in [OWSM Policies Supported for Identity Context](#). For details about creating a new policy from a predefined policy, see [Cloning a Web Service Policy](#). Once you have created the policies, you can edit them and set the `propagate.identity.context` property as described in this section.

For information about overriding the `propagate.identity.context` property when you attach the policy, see [Overriding Policy Configuration Properties](#).

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 the `propagate.identity.context` to `true`.

To configure identity propagation using a cloned policy:

1. In the Target Navigation pane, expand **WebLogic Domain** and select the domain for which you need to configure identity context propagation.
2. From the **WebLogic Domain** menu, select **Web Services**, then **WSM Policies**.
3. Select the cloned SAML policy for which you want to enable identity context propagation and click **Open**.

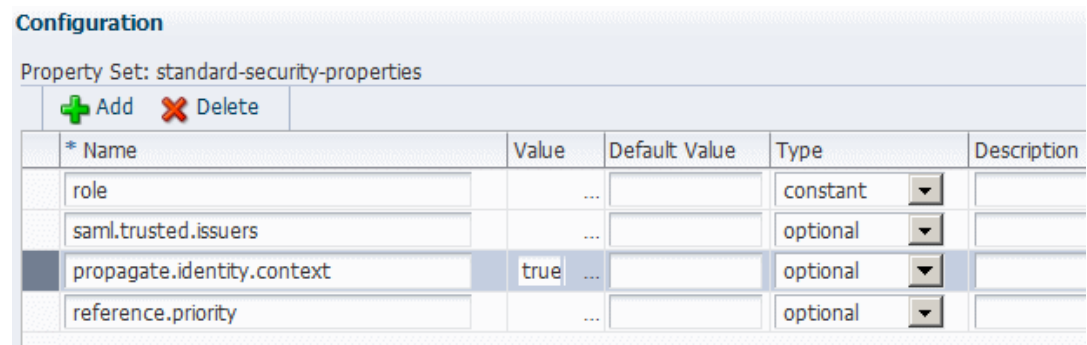
Remember that you have to enable identity context propagation for both the web service client and web service policies.

4. Select the **Assertions** tab.
5. Click **Configuration**.

The Configuration page displays a list of the configuration properties for the policy.

6. Enter `true` in the **Value** field for the `propagate.identity.context` property.
7. Click **OK**.

Figure 11-6 Setting the `propagate.identity.context` Property to True



Configuration					
Property Set: standard-security-properties					
		<input type="button" value="+ Add"/> <input type="button" value="X Delete"/>			
* Name	Value	Default Value	Type	Description	
role	...		constant		
saml.trusted.issuers	...		optional		
propagate.identity.context	true	...	optional		
reference.priority	...		optional		

8. Click **Save** to submit the changes.
9. Repeat Steps 3 and 8 for the corresponding client or service policy, as appropriate.
10. Attach the policy to the desired endpoints as described in [Attaching Policies to Manage and Secure Web Services](#).

11.6 Understanding Kerberos Token Configuration

An introduction to Kerberos Token Configuration is detailed in this section.

For information on the Kerberos authentication protocol, see "Understanding the Kerberos Protocol" in *Understanding Oracle Web Services Manager*.

The predefined Kerberos policies include `kerberos` in their name and are listed in [Oracle Web Services Manager Predefined Policies](#). The predefined Kerberos assertion templates include `kerberos` in their name and are listed in [Oracle Web Services Manager Predefined Assertion Templates](#).

Follow the steps described in these sections to configure Kerberos for use by the web service client and web service.

- [About MIT Kerberos](#)
- [About Using Microsoft Active Directory with Key Distribution Center](#)
- [Setting the Service Principal Name In the Web Service Client](#)
- [Configuring the Web Service to Use the Correct KDC](#)
- [About Using the Correct Keytab File in Enterprise Manager](#)
- [Authenticating the User Corresponding to the Service Principal](#)
- [Creating a Ticket Cache for the Web Service Client](#)
- [Kerberos Configuration Over SSL](#)
- [Kerberos Configuration with SPNEGO Negotiation](#)
- [About Configuration of Credential Delegation](#)

11.6.1 About MIT Kerberos

You can use MIT Kerberos as your KDC.

This section describes the following tasks when using MIT Kerberos as your KDC:

- [Initializing and Starting the MIT Kerberos KDC](#)
- [Creating Principals](#)
- [Configuring the Web Service Client to Use the Correct KDC](#)

11.6.1.1 Initializing and Starting the MIT Kerberos KDC

This section describes how to initialize and start the MIT Kerberos KDC.

Initialize 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 &
```

11.6.1.2 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.

Create two accounts in the KDC user registry. 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 the Kerberos System. Using a random password increases security.

11.6.1.3 Configuring the Web Service Client to Use the Correct KDC

You must 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.

A sample `krb5.conf` is shown in the example in this section. Note the following:

- The file tells the kerberos runtime 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_encypes`
- `default_tgs_encypes`
- `permitted_encypes`

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_encypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc arcfour-hmac`
 - * `default_tgs_encypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc arcfour-hmac`
 - * `permitted_encypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc arcfour-hmac`
- `wss11_kerberos_with_message_protection_basic128_client_policy`:
 - * `default_tkt_encypes = arcfour-hmac des3-cbc-sha1 des-cbc-md5 des-cbc-crc`
 - * `default_tgs_encypes = arcfour-hmac des3-cbc-sha1 des-cbc-md5 des-cbc-crc`
 - * `permitted_encypes = arcfour-hmac des3-cbc-sha1 des-cbc-md5 des-cbc-crc`

The following example shows a 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 = someadminserver.com:88 admin_server = someadminserver.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 }
```

11.6.2 About Using Microsoft Active Directory with Key Distribution Center

You can use Microsoft Active Directory with 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.

 **Note:**

When using versions of Microsoft Active Directory earlier than 2008 as the KDC in message protection scenarios, do not use the following policies:

- `oracle/wss11_kerberos_token_with_message_protection_client_policy`
- `oracle/wss11_kerberos_token_with_message_protection_service_policy`

These policies use Triple DES encryption, which is not supported by earlier versions of Microsoft Active Directory.

Instead, use the Kerberos message protection policies that include "basic128" in the policy name, specifically:

- `oracle/wss11_kerberos_token_with_message_protection_basic128_client_policy`
- `oracle/wss11_kerberos_token_with_message_protection_basic128_service_policy`

These two policies are compatible with all supported versions of Microsoft Active Directory.

This section describes the following topics:

- [Web Service Client Set Up Tasks](#)
- [Setting Up the Web Service](#)

11.6.2.1 Web Service Client Set Up Tasks

This section describes how to set up the Web Service Client. It includes the following tasks:

- [Creating a User Account](#)
- [Setting the Service Principal Name](#)
- [Creating a Keytab File](#)

11.6.2.1.1 Creating a User Account

This section describes how to 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 `group/name`.

11.6.2.1.2 Setting the Service Principal Name

This section describes how to 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>`.

11.6.2.1.3 Creating a Keytab File

This section describes how to 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`.

11.6.2.2 Setting Up the Web Service

This section describes how to set up the Web Service.

Perform the following 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 [Creating a Keytab File](#) 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 Kerberos Login Module](#) 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
```

11.6.3 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.

This section describes the steps the set up the service principal name.

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`.

Alternatively, you can specify the service principal name at design time by using a configuration override; for example:

```
JAX-WS Clients:
((BindingProvider)port).getRequestContext().put(SecurityConstants.ClientConstants.WSS_KER
BEROS_SERVICE_PRINCIPAL,
SOAP/myhost.example.com@example.com);
```

11.6.4 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 the example in [Configuring the Web Service Client to Use the Correct KDC](#). This example also applies to the web service KDC configuration.

11.6.5 About Using the Correct Keytab File in Enterprise Manager

You should use the correct keytab file in Enterprise Manager.

To use the correct keytab file, you

- Extract and install the keytab File
- Export the keytab file
- Modify the krb5 login module

These tasks are described in the following sections:

- [Extracting the Keytab File](#)
- [Exporting the Keytab File](#)
- [Modifying the krb5 Login Module to use the Keytab File](#)

11.6.5.1 Extracting the Keytab File

This section describes the procedure to extract the keytab file.

You need to 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
```

11.6.5.2 Exporting the Keytab File

This section describes the procedure to export the keytab file.

You must then 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.

11.6.5.3 Modifying the krb5 Login Module to use the Keytab File

This section describes the procedure to modify the krb5 module to use the keytab file.

To modify the krb5 login module as described in [Configuring the Kerberos Login Module](#) 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`

11.6.6 Authenticating the User Corresponding to the Service Principal

The web services runtime 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. For more information about authentication providers, see [Supported Authentication Providers in WebLogic Server](#).

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.

11.6.7 Creating a Ticket Cache for the Web Service Client

You can create a ticket cache for the web service client.

To create a ticket cache for the web service client, perform the following steps:

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.

11.6.8 Kerberos Configuration Over SSL

The two basic assertion templates for Kerberos, `oracle/wss11_kerberos_token_client_template` and `oracle/wss11_kerberos_token_service_template`, provide authentication using Kerberos tokens but do not provide any form of message protection.

OWSM provides the following assertion templates that add message protection using SSL transport-level security to the basic authentication using Kerberos tokens:

- [oracle/wss11_kerberos_token_over_ssl_client_template](#)
- [oracle/wss11_kerberos_token_over_ssl_service_template](#)

To use these assertion templates, you need to configure SSL, as described in [About Configuring Keystores for SSL](#).

These assertions use the Kerberos token for authentication and for signing the timestamp, and rely on underlying One-Way SSL transport for message protection. They advertise the Kerberos token as an `EndorsingSupportingToken`.

11.6.9 Kerberos Configuration 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.

For more information on using Kerberos with SPNEGO, see "Kerberos and SPNEGO" in *Understanding Oracle Web Services Manager*.

OWSM 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.

You can use these assertion templates to create a new policy, or to add a SPNEGO assertion to an existing multi-token policy. For more information about creating policies or adding an assertion to a policy, see the following sections:

- [Creating a New Web Service Policy](#)
- [Adding Assertions to a Policy](#)

11.6.10 About Configuration of Credential Delegation

Kerberos includes a feature called credential delegation to support scenarios where a client request demands that one service use a second service (on behalf of the client) to satisfy the request. To satisfy such scenarios, the client provides the first service with a forwarded TGT

(ticket granting ticket) that the first service uses to obtain a service ticket for the second service from the KDC.

For more information on credential delegation, see "Credential Delegation in Kerberos" in *Understanding Oracle Web Services Manager*.

To configure credential delegation, you must perform these tasks:

- Configure Kerberos to allow credential delegation.
- Configure the domain where the client is deployed to allow OWSM to forward a Kerberos ticket to the service on behalf of the client.
- Enable credential delegation in the client policy and the service policy.

When using credential delegation with a service in active directory, you must also configure the service's SPN account to be trusted for delegation.

The following subsections describe these tasks:

- [Configuring Credential Delegation in Kerberos](#)
- [Configuring Delegation Permission to OWSM](#)
- [Enabling Credential Delegation in the Client Policy and the Service Policy](#)
- [Configuring Credential Delegation for a Service in Active Directory](#)
- [Configuring Credential Delegation With An Example](#)

11.6.10.1 Configuring Credential Delegation in Kerberos

This procedure describes how to configure credential delegation in kerberos.

To configure Kerberos to support credential delegation, you must set the `forwardable` option to `true` in the `libdefaults` section of the `krb5.conf` file; for example:

```
[libdefaults]
forwardable = true
```

11.6.10.2 Configuring Delegation Permission to OWSM

In order to delegate credentials, OWSM must have permission to forward a Kerberos ticket to the service on behalf of the client.

To provide the permission, you use a `grantPermission` WLST command of this form:

```
grantPermission(
  codeBaseURL="file:${common.components.home}/modules/oracle.wsm.common_12.1.2/wsm-agent-core.jar",
  permClass="javax.security.auth.kerberos.DelegationPermission",
  permTarget="\service_principal@REALM_NAME\" \"krbtgt/REALM_NAME@REALM_NAME\"")
```

In this command, `service_principal` is the service principal and `REALM_NAME` is the security realm.

Because `grantPermission` is an online command, you must first use the `connect` WLST command to connect to the WebLogic Server instance. For more information about the `grantPermission` command, see "grantPermission" in *Infrastructure Security WLST Command Reference*.

11.6.10.3 Enabling Credential Delegation in the Client Policy and the Service Policy

This procedure shows how to enable credential delegation in the client policy and service policy.

You enable credential delegation by setting the value of the `credential.delegation` configuration property to `true` in the Kerberos assertions for the client policy and the service policy. This property is available in all of the Kerberos assertion templates. By default, this value is `false`, indicating that credential delegation is disabled.

11.6.10.4 Configuring Credential Delegation for a Service in Active Directory

When using Kerberos credential delegation with a service in active directory, you must configure the service's SPN domain account to be trusted for delegation.

1. On the domain controller, start Active Directory Users and Computers.
2. In the left pane, click **Users**.
3. In the right pane, right-click the name of the domain account you used when creating the service's SPN, and then click **Properties**.
4. On the Delegation tab, select the Account is trusted for delegation option, and then click **OK**.
5. Quit Active Directory Users and Computers.



Note:

Java clients that use the standard JDK libraries for Kerberos ignore this setting in Active Directory due to a limitation in the JDK.

11.6.10.5 Configuring Credential Delegation With An Example

This procedure explains how to configure credential delegation.

The following example demonstrates how to configure credential delegation using the following scenario:

A bank provides its customers a client application, MyBanker, that enables account holders to view information about their accounts. To get transaction information for an account, the MyBanker client calls the Transactions service. This service has direct access to transactions for an account since the account's last statement. To get information about older transactions, it calls the OldTransactions service. To get information from OldTransactions, Transactions must authenticate with the credentials of the MyBanker client.

Prerequisites

This example requires that the MyBanker client and the Transactions and OldTransactions services all authenticate using the same Kerberos KDC.

Procedure

In the following procedure, Steps 2, 3 and 4 are very similar to those that you would routinely perform when connecting clients and services using OWSM. Steps 1 and 5 are unique to credential delegation.

1. Configure the domain where MyBanker is deployed to permit OWSM to forward tickets to Transactions on behalf of MyBanker by adding a `<grant>` element to the `system-jazn-data.xml` file, as described in [Configuring Delegation Permission to OWSM](#).
2. Attach the appropriate Kerberos or SPNEGO **service** policy to the Transactions service. Configure the policy as you normally would; additionally:
 - Set the `credential.delegation` property to `true`.
3. Attach the corresponding Kerberos or SPNEGO **client** policy to the MyBanker client. Configure the policy as you normally would; additionally:
 - Set the `service.principal.name` property to the principal name of the Transactions service.
 - Make sure the `caller.principal.name` and `keytab.location` properties are set.
 - Set the `credential.delegation` property to `true`.
4. Attach the appropriate predefined Kerberos or SPNEGO **service** policy to the OldTransactions service. Configure the policy as you normally would.
5. Attach the corresponding predefined Kerberos or SPNEGO **client** policy to the Transactions service. Configure this policy as follows:
 - Set the `service.principal.name` property to the principal name of the OldTransactions service.
 - Make sure the `caller.principal.name` and `keytab.location` properties are **not** set.

To extend this example to accommodate additional delegations of MyBanker's credentials, you would need to perform these tasks for each additional delegation:

- Give OWSM permission to forward tickets to the service receiving the delegated credentials as described in [Configuring Delegation Permission to OWSM](#).
- Set the `credential.delegation` property to `true` in the service policy for the service forwarding the delegated credentials.
- Add a service policy to the service receiving the delegated credentials.
- Add the corresponding client policy to the service forwarding the delegated credentials, configuring it as described in Step 5 of the example procedure.

11.7 About WS-Trust Configuration

You can configure and use the OWSM policies for WS-Trust using Security Token Services (STSeS).

The first subsection, [Overview of Web Services WS-Trust](#), provides introductory information about WS-Trust and how it uses STSeS to broker trust between web service clients and web services. The remaining subsections describe specific configuration topics related to WS-Trust:

- [Overview of Web Services WS-Trust](#)
- [Supported STS Servers](#)
- [Understanding Token Lifetime and Token Caching](#)
- [Setting Up Automatic Policy Configuration for STS](#)
- [About Configuring Web Services Federation](#)
- [Overview of SAML Holder of Key and SAML Bearer as Issued Tokens](#)
- [Understanding SAML Sender Vouches as Issued Tokens](#)

- Overview of **On Behalf Of** Use Cases
- Programmatically Overriding Policy Configuration for WS-Trust Client Policies
- Overview of JWK Document Trust Configuration
- About IDCS Discovery Service
- About Token Audience Configuration

11.7.1 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 Security Token Service (STS), which acts as trust broker. That is, the web service client and the web service do not explicitly trust each other; instead, they implicitly trust each other because they both trust the STS.

When using an STS as a trust broker, authentication proceeds as follows:

1. The web service client authenticates itself to the STS and requests a security token to be issued to it. This is called the RequestSecurityToken (RST) message.
2. The STS confirms the web service client's credentials and then responds to the security token request. This is called the RequestSecurityTokenResponse (RSTR) message.
3. The web service client presents to the web service the security token that the STS issued to it.
4. The web service confirms the validity of the issued security token.

Upon successful completion of this process, the web service client can make requests to the web service.

When OWSM is used to manage authentication, the actual process to achieve the preceding generic process is as follows:

1. The web service client wants to invoke a web service. The OWSM agent attempts to fetch the WSDL of the web service and extract the issued token service policy. The OWSM 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 OWSM agent issues an RST to the STS.
3. The STS responds with an RSTR.
4. The OWSM agent processes the RSTR sent by the STS and propagates the issued token to the web service.
5. The OWSM agent servicing the web service processes and verifies the issued token and generates a response back.

Web Services Federation

In acting as a trust broker, an STS provides two main services:

- It issues tokens to authenticated web service clients
- It validates tokens as requested by web services

Additionally, an STS can accept a token issued by another STS and issue its own token based on the received token's content, provided that it trusts the other STS. This ability of STSes to trust each other and to transform issued tokens enables federated trust across security domains, which is called web services federation.

As an example, a web service client in one security domain wants to invoke a web service that is in another security domain. If there is an STS in each domain, and they trust each other, the web service client can invoke the web service as follows:

1. The web service client authenticates itself to the STS in its security domain and requests a security token to be issued to it. This STS is called the Identity Provided STS, or IP-STS.
2. The IP-STS confirms the web service client's credentials and then responds to the security token request, providing an issued token.
3. The web service client presents the issued token from the IP-STS to the STS in the web service's security domain and requests a security token be issued to it. This second STS is called the Relying Party STS, or RP-STS.
4. The RP-STS confirms the validity of the issued token from the IP-STS. Using data from the IP-STS's token, it creates its own token and provides it to the web service client.
5. The web service client presents to the web service the issued token from the RP-STS.
6. The web service confirms the validity of the issued token from the STS that it trusts, the RP-STS in its own security domain.

The preceding process shows how a web service client can communicate with a web service in a different security domain due to the trust shared between STSes in the two domains. However, this shared trust between STSes can be extended even further: if the IP-STS and the RP-STS do not trust each other directly, but share trust with the same intermediary STS, a chain of trust can be formed from the IP-STS to the intermediary STS to the RP-STS. Moreover, additional intermediary STSes can be positioned between the IP-STS and the RP-STS, provided that a chain of trust can be traced from the IP-STS to the RP-STS.

For information about using OWSM to configure web services federation, see [About Configuring Web Services Federation](#).

11.7.1.1 Understanding the Mechanism to Obtain STS Configuration

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](#)) 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 OWSM repository. (For details about the repository, see [Managing the Oracle Web Services Manager 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 OWSM 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.

11.7.1.2 Understanding the Token Types 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
- SAML Holder of Key Asymmetric
- SAML Bearer

For information about these tokens, see [Overview of SAML Holder of Key and SAML Bearer as Issued Tokens](#).

An STS can also return SAML Sender Vouches tokens. For more information, see [Understanding SAML Sender Vouches as Issued Tokens](#).

11.7.2 Supported STS Servers

OWSM provides a standard WS-Trust client. This client has been certified to interoperate with the following Security Token Services.

- Oracle Security Token Service (OSTS)

- Oracle OpenSSO STS
- Microsoft ADFS 2.0 Security Token Service (STS)

11.7.3 Understanding Token Lifetime and Token Caching

Token lifetime and token caching is detailed in this section.

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, OWSM validates the timestamp and rejects the message if the response has expired.

Additionally, OWSM supports caching of returned tokens. The configuration property `issued.token.lifetime` in issued token client policies control caching of returned tokens.

When making a request to STS, OWSM requests a lifetime for returned tokens for the period specified by `issued.token.lifetime` in the RST request message to an STS.

If the STS returns a token lifetime value different from the requested `issued.token.lifetime` value, OWSM uses the return value as the period for caching returned tokens. If the STS returns an empty token expiry value, OWSM does not cache returned tokens.

Thus, the `issued.token.lifetime` property is used only to request a particular lifetime for returned tokens. The STS is not obligated to honor the value, and OWSM performs token caching based on the actual lifetime value returned by the STS.

If you do not provide a value for `issued.token.lifetime`, OWSM uses the domain default value of 28800000 milliseconds (eight hours). To change this domain default value, see [Configuring the Lifetime for the Issued Token Using Fusion Middleware Control](#).

11.7.4 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, OWSM 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:

- [Understanding the Requirements for Automatic Policy Configuration](#)
- [Main Steps in Setting Up Automatic Policy Configuration](#)
- [Manually Configuring the STS Config Policy From the Web Service Client: Main Steps](#)

11.7.4.1 Understanding the 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](#).
- 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 [Overview of Configuring Keystores for Message Protection](#) for information on how to do this.
- The client's public key must be available in the STS keystore.

11.7.4.2 Main Steps in Setting Up Automatic Policy Configuration

This section describes the tasks that you need to perform for setting up Automatic Policy Configuration.

This section includes the following topics:

- [Configuring a Policy for Automatic Policy Configuration](#)
- [Configuring a Web Service Client for Automatic Policy Configuration](#)
- [Configuring a Web Service for Automatic Policy Configuration](#)

11.7.4.2.1 Configuring a Policy for Automatic Policy Configuration

This section describes how to configure a policy for automatic policy configuration.

Perform the following 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 OWSM keystore.
2. Optionally, add the Distinguished Name of the STS to the Trusted STS list, as described in [Understanding Trusted Issuers and Trusted Distinguished Names List for SAML Signing Certificates](#).
3. If you want to use SAML HOK symmetric, you need to add an entry in the 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 Port URI setting 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.

11.7.4.2.2 Configuring a Web Service Client for Automatic Policy Configuration

This section describes how to configure a web service client for automatic policy configuration.

Perform the following 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 [Predefined Assertion Templates for Oracle Web Services](#) for descriptions of these properties.

- `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 [Overview of Configuring Keystores for Message Protection](#) for information on how to do this.

11.7.4.2.3 Configuring a Web Service for Automatic Policy Configuration

This section describes how to configure a web service for automatic policy configuration.

Perform the following steps to 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.csrf.key` in the issued-token service policy, if required.
4. Make sure the client's public key is available in the OWSM keystore.

11.7.4.3 Manually Configuring the STS Config Policy From the Web Service Client: Main Steps

You should configure the STS config policy from the web service by setting up automatic policy for STS. However, in certain situations, automatic policy configuration doesn't work, and you must configure it manually from the web service client.

These situations are:

- You did not configure the STS config policy from the web service.
- You are using the SAML sender vouches confirmation method.
- You are using a JSE client. Automatic Policy Configuration does not work with JSE clients.

To configure the STS config policy from the web service client using Fusion Middleware Control:

1. Attach both web service client and config policy to your web service client. You must attach the policies in this order:
 - The STS trust config client policy. i.e. `oracle/sts_trust_config_client_policy`
 - Issued token client policy

Note:

If you attach multiple instances of the `sts_trust_config_client_policy`, no error is generated. However, only one instance is enforced, and you cannot control which instance that is.

2. Override the following properties depending on the use case. See [Predefined Assertion Templates for Oracle Web Services](#) for descriptions of these properties.
 - `BindingProvider.USERNAME_PROPERTY(javax.xml.ws.security.auth.username)`
 - `BindingProvider.PASSWORD_PROPERTY(javax.xml.ws.security.auth.password)`
 - `oracle.wsm.security.util.SecurityConstants.ClientConstants.ATTESTING_MAPPING_ATTRIBUTE`
 - `oracle.wsm.security.util.SecurityConstants.ClientConstants.CALLER_PRINCIPAL_NAME`
 - `oracle.wsm.security.util.SecurityConstants.ClientConstants.ON_BEHALF_OF`
 - `oracle.wsm.security.util.SecurityConstants.ClientConstants.SAML_AUDIENCE_URI`

- oracle.wsm.security.util.SecurityConstants.ClientConstants.STS_KEYSTORE_RECIPIENT_ALIAS
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_INCLUDE_USER_ROLES
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_ISSUED_TOKEN_LIFETIME
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_PASSWORD_PROPERTY
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_SAML_ASSERTION_FILE_NAME
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_STS_AUTH_ON_BEHALF_OF_CSF_KEY
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_STS_AUTH_USER_CSF_KEY
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_STS_AUTH_X509_CSF_KEY
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_SUBJECT_PRECEDENCE
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_USERNAME_PROPERTY
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_CSF_KEY
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_ENC_KEY_ALIAS
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_ENC_KEY_PASSWORD
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KERBEROS_SERVICE_PRINCIPAL
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_LOCATION
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_PASSWORD
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_TYPE
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_RECIPIENT_KEY_ALIAS
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_SAML_ISSUER_NAME
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_SIG_KEY_ALIAS
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_SIG_KEY_PASSWORD
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_WSDL_URI
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_PORT_URI
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_PORT_ENDPOINT
- oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_POLICY_REFERENCE_URI

3. Save your changes.

11.7.5 About Configuring Web Services Federation

You have to configure web services, web service clients and STSes for web services federation.

The following sections describe how to configure web services, web service clients and STSes for web services federation:

- [Configuring a Web Service for Web Services Federation](#)
- [About Configuring a Web Client for Web Services Federation](#)
- [Configuring an STS for Web Services Federation](#)

11.7.5.1 Configuring a Web Service for Web Services Federation

This section describes how to configure web services for web services federation.

Perform the following steps:

1. Attach the desired issued token service policy to the service. OWSM provides the following predefined policies:
 - `oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy`
 - `oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy`
2. Optionally, attach the `oracle/sts_trust_config_service_policy` policy and configure it to refer to the RP-STS. Attaching and configuring this policy enables automatic policy configuration for the RP-STS, which permits any web service clients using the service to avoid having to attach an `oracle/sts_trust_config_client_policy` policy for the RP-STS. For more information, see [Setting Up Automatic Policy Configuration for STS](#).
3. Import the signing certificate for the RP-STS into the OWSM keystore.
4. Define the RP-STS as a trusted issuer and a trusted DN, as described in [Understanding Trusted Issuers and Trusted Distinguished Names List for SAML Signing Certificates](#).

11.7.5.2 About Configuring a Web Client for Web Services Federation

How you configure a web client for web services federation depends on whether issuer information is provided in the WSDLs for the web service and for the RP-STS.

It contains the following scenarios:

- [Configuring the Web Client When Issuer Is in the Service WSDL and Is Not in the RP-STS WSDL](#)
- [Configuring the Web Client When Issuer Is Not in the Service WSDL and Is Not in the RP-STS WSDL](#)
- [Configuring the Web Client When Issuer Is in the Service WSDL and Is in the RP-STS WSDL](#)
- [Configuring the Web Client When Issuer Is Not in the Service WSDL and Is in the RP-STS WSDL](#)

11.7.5.2.1 Configuring the Web Client When Issuer Is in the Service WSDL and Is Not in the RP-STS WSDL

This is the most common use case, and is the use case when the web service is configured for automatic policy configuration of the RP-STS.

For this use case, perform these steps to configure the web client:

1. Attach the issued token client policy corresponding to the issued token service policy that is attached to the web service and configure it to refer to the web service. In addition, if you are using the message protection policy, set the `sts.keystore.recipient.alias` configuration property to the certificate alias of the RP-STS; if you are using the SSL policy, do not set this property.
2. Attach the `oracle/sts_trust_config_client_policy` policy and configure it to refer to the IP-STS.

11.7.5.2.2 Configuring the Web Client When Issuer Is Not in the Service WSDL and Is Not in the RP-STS WSDL

This is a less common use case, and is the use case when the web service is not configured for automatic policy configuration of the RP-STS.

For this use case, perform these steps configure the web client:

1. Attach the issued token client policy corresponding to the issued token service policy that is attached to the web service and configure it to refer to the web service. Additionally, set the `sts.in.order` configuration property to the endpoint URI of the RP-STS followed by the endpoint URI of the IP-STS. Separate the two URIs with a semicolon (;). For example:

```
http://m2.example.com:14100/sts/wssbearer;  
http://http://m1.example.com/adfs/services/trust/13/usernamemixed
```

2. Attach the `oracle/sts_trust_config_client_policy` policy and configure it to refer to the IP-STS.
3. Attach the `oracle/sts_trust_config_client_policy` policy and configure it to refer to the RP-STS.

11.7.5.2.3 Configuring the Web Client When Issuer Is in the Service WSDL and Is in the RP-STS WSDL

This is a rare use case, and is the use case when the web service is configured for automatic policy configuration of the RP-STS.

For this use case, perform these steps configure the web client:

1. Attach the issued token client policy corresponding to the issued token service policy that is attached to the web service and configure it to refer to the web service. In addition, if you are using the message protection policy, set the `sts.keystore.recipient.alias` configuration property to the certificate alias of the RP-STS; if you are using the SSL policy, do not set this property.
2. Attach the `oracle/sts_trust_config_client_policy` policy and configure it to refer to the IP-STS.

11.7.5.2.4 Configuring the Web Client When Issuer Is Not in the Service WSDL and Is in the RP-STS WSDL

This is a rare use case.

For this use case, perform these steps configure the web client:

1. Attach the issued token client policy corresponding to the issued token service policy that is attached to the web service and configure it to refer to the web service. Additionally, if you are using the message protection policy, set the `sts.in.order` configuration property to the endpoint URI of the RP-STS followed by the endpoint URI of the IP-STS; if you are using the SSL policy, do not set this property.
2. If you are using the message protection policy, attach the `oracle/sts_trust_config_client_policy` policy and configure it to refer to the IP-STS; if you are using the SSL policy, do not perform this step.
3. Attach the `oracle/sts_trust_config_client_policy` policy and configure it to refer to the RP-STS.

11.7.5.3 Configuring an STS for Web Services Federation

This section describes how to configure an STS for Web Services Federation.

The following steps to configure an STS as an RP-STS or an IP-STS are general, high-level steps. For detailed steps, refer to the documentation for the particular STS. For Oracle STS, see <http://www.oracle.com/technetwork/middleware/id-mgmt/overview/oraclests-166231.html>.

For Microsoft ADFS 2.0 STS, see [http://technet.microsoft.com/en-us/library/adfs2\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/adfs2(v=ws.10).aspx).

To configure the RP-STS for web services federation, use the administrative interface of the RP-STS to perform these steps:

1. Add the web service as a relying party.
2. Add the IP-STS as a trusted identity provider.

In Oracle STS, a trusted identity provider is called an Issuing Authority.

In Microsoft ADFS 2.0 STS, a trusted identity provider is called a Claim Provider.

To configure the IP-STS for web services federation, use the administrative interface of the IP-STS to perform this step:

- Add the RP-STS as a relying party.

11.7.6 Overview of SAML Holder of Key and SAML Bearer as Issued Tokens

In general, an STS returns tokens in the SAML assertion.

An STS generally returns one of the following kinds of tokens in the SAML assertion:

- 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 [Determining the Proof Key for SAML HOK Only](#).

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.

11.7.6.1 Determining the Proof Key for 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` policy in the `<key-type>` entry in the `<sp:IssuedToken>` policy assertion. For example,

```
<orasp:request-security-token-templateorasp: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>

This section contains the following topics:

- [About Symmetric Proof Key](#)
- [About Asymmetric Proof Key](#)

11.7.6.1.1 About 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 OWSM 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 OWSM 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 OWSM agent uses the STS entropy as the key material for signing and encryption.

11.7.6.1.2 About 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`.

11.7.7 Understanding SAML Sender Vouches as Issued Tokens

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 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. However, 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.

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](#)) cannot be used for SAML sender vouches because the web service WSDL will not have information about the STS.

11.7.8 Overview of *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 [Oracle Web Services Manager Predefined Assertion Templates](#). 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.

11.7.9 Programmatically Overriding Policy Configuration for WS-Trust Client Policies

You can set properties via programmatic configuration overrides for a given policy.

[Table 5-1](#) shows the properties you can set via programmatic configuration overrides for a given policy.

The following list shows how to override STS properties programmatically for a set of sample use cases.

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.

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.

11.7.10 Overview of JWK Document Trust Configuration

OWSM has support for JSON Web Key (JWK). JWK provides a dynamic approach where tokens can be validated in real time without prior import of the keys or certificates. It helps in scenarios where a number of external identity providers periodically rotate their token signing keys as a security measure and OWSM can automatically uptake those changes dynamically.

JSON Web Key (JWK) and JWK Set

JWK is a JSON object that represents a cryptographic key. The members of the object represent properties of the key, including its value. JWK Set is a JSON object that represents a set of JWKs. The JSON object MUST have a "keys" member, which is an array of JWKs.

Parameter	Description
<code>kty</code>	Key Type. It identifies the cryptographic algorithm family used with the key, such as RSA.
<code>alg</code>	Algorithm. It identifies the algorithm to be used with the key.
<code>use</code>	Public key use. It identifies the intended use of the public key. Values are: <ul style="list-style-type: none"> • <code>sig</code>: Signature • <code>enc</code>: Encryption
<code>kid</code>	Key ID. It is used to match a specific key, for instance, to choose among a set of keys within a JWK Set during key rollover.
<code>x5u</code>	X.509 URL. It is a URI that refers to a resource for an X.509 public key certificate or certificate chain.
<code>x5c</code>	X.509 Certificate Chain. It contains a chain of one or more PKIX certificates. The certificate chain is represented as a JSON array of certificate value strings. Each string in the array is a base64-encoded DER PKIX certificate value.
<code>x5t</code>	X.509 certificate SHA-1 thumbprint. It is a base64url-encoded SHA-1 thumbprint of the DER encoding of an X.509 certificate.
<code>x5t#\$256</code>	X.509 Certificate SHA-256 Thumbprint. It is a base64url-encoded SHA-256 thumbprint of the DER encoding of an X.509 certificate.
<code>n</code>	Modulus. It is the modulus part of the public key.
<code>e</code>	Exponent. It is the exponent part of the public key.

See Also:

- Token Issuer Trust Configuration Commands in *WLST Command Reference for Infrastructure Components* for CLI commands to import and revoke configuration from JWK document.
- Manage Token Issuer Trust Configurations in *REST API for Managing Credentials and Keystores with Oracle Web Services Manager* for REST API methods to import and revoke configuration from JWK document.

11.7.11 About IDCS Discovery Service

OWSM supports dynamic open id connect discovery to configure trust.

Discovery service provides a dynamic approach where a token is validated in real time without prior import of the keys or certificates which are used to sign tokens to configure trust . OWSM leverages OpenID provider metadata to configure trust dynamically.

IDCS publishes its metadata document at two endpoints below:

- `<base-url>/well-known/idcs-configuration` (public and proprietary)
- `<base-url>/well-known/openid-configuration` (public and standard)

See Also:

- Token Issuer Trust Configuration Commands in *WLST Command Reference for Infrastructure Components* for CLI commands to import and revoke configuration.
- Manage Token Issuer Trust Configurations in *REST API for Managing Credentials and Keystores with Oracle Web Services Manager* for REST API methods to import and revoke configuration.

11.7.12 About Token Audience Configuration

OWSM supports token audience configuration in order to validate external tokens used by external identity providers differently.

OWSM validates a token audience, by ensuring that at least one of the audience values specified within the token matches, (or is a prefix of) the URL being called. In some cases, an external identity provider may use the audience claim/attribute differently, in such scenarios OWSM has the ability to specify one or more audience values which override the default audience values.

This override of the default audience value is done by configuring one or more audience values. OWSM considers the token to be valid if the audience value list in the token contains at least one value which matches any one of the configured values.

Token Audience Configuration helps to support Mobile Cloud Service (MCS) inter-work with external identity providers.

See Also:

- Token Issuer Trust Configuration Commands in *WLST Command Reference for Infrastructure Components* for CLI commands to import and revoke configuration.
- Import TrustDocument Name Configurations Method in *REST API for Managing Credentials and Keystores with Oracle Web Services Manager* for REST API methods to import and revoke configuration.

12

Configuring Secure Conversation Using Oracle Web Services Manager

This chapter describes how OWSM implements the Web Services Trust (WS-Trust 1.3) and Web Services Secure Conversation (WS-SecureConversation 1.3) specifications, which together provide secure communication between web services and their clients. You can use WS-SecureConversation to increase the performance and security of your web services. This chapter includes the following sections:

- [Overview of Web Services Secure Conversation Language Specification](#)
- [About Configuring Secure Conversation](#)
- [Attaching a Secure Conversation Policy at Design Time](#)
- [About Configuring Persistence](#)
- [Understanding Secure Conversation Sessions](#)

12.1 Overview of Web Services Secure Conversation Language Specification

The Web Services Secure Conversation Language (WS-SecureConversation) specification defines extensions that build on Web Services Security (WS-Security) 1.1 and 1.0 and Web Services Trust Language (WS-Trust) to provide secure communication across one or more messages. Specifically, this specification defines mechanisms for establishing and sharing security contexts, and deriving keys from established security contexts (or any shared secret).

Link to the specification document: <http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/os/ws-secureconversation-1.4-spec-os.doc>

A particularly important use of WS-SecureConversation is to provide security for WS-ReliableMessaging (WS-RM) policies, as described in "Using WS-SecureConversation With WS-ReliableMessaging" in *Understanding Oracle Web Services Manager*. When you attach both a WS-SecureConversation policy and a WS-RM policy, the WS-RM policy benefits from the use of secure conversation to prevent sequence attacks.

You can also attach a WS-SecureConversation policy to an asynchronous web service either with or without a WS-RM policy. The following combinations are possible:

- WS-RM policy and a WS-SecureConversation policy
- Asynchronous web service, a WS-RM policy, and a WS-SecureConversation policy

For more information on secure conversation and when you might want to use it, see "Understanding Secure Conversation" in *Understanding Oracle Web Services Manager*.

12.2 About Configuring Secure Conversation

In most cases, you do not need to perform any WS-SecureConversation-specific configuration other than creating a policy and enabling secure conversation.

If you use the preconfigured WS-SecureConversation policies described in [OWSM Policies Supported for WS-SecureConversation](#), WS-SecureConversation is already enabled. All of the other configuration that is needed is the same as for the policy without WS-SecureConversation.

If you configure WS-SecureConversation, you must configure it for both the web service and client policies. This is because the OWSM client and server agents participate in the handshake process to establish the Security Context Token (SCT) used to secure the conversation. The OWSM client sends the RST (Request Security Token) to the service, which then returns the RSTR (Request Security Token Response) containing the SCT.

The OWSM security policies described in [OWSM Policies Supported for WS-SecureConversation](#), include a configuration setting that allows you to enable and configure WS-SecureConversation for that policy. You may find that using these preconfigured policies makes your security tasks easier to view at a glance and manage.

In addition, policies based on many of the predefined assertion templates also support WS-SecureConversation. See [Oracle Web Services Manager Predefined Assertion Templates](#) for additional information.

This section contains the following topics:

- [Configuring Secure Conversation Using Fusion Middleware Control](#)
- [Configuring Secure Conversation Using WLST](#)

12.2.1 Configuring Secure Conversation Using Fusion Middleware Control

This topic describes the steps that you need to perform to configure WS-SecureConversation using Fusion Middleware Control.

To configure the WS-SecureConversation:

1. Optionally, configure WS-Secure Conversation at the domain level, as described in [Secure Conversation Configuration for the Domain Using Fusion Middleware Control](#). You need to do this only if the defaults do not meet your needs.
2. In the content pane, click **WebLogic Domain**, then **Web Services**, and then **Policies**.
3. Select one of the preconfigured WS-SecureConversation policies, or another security policy for which you want to enable WS-SecureConversation from the list in [OWSM Policies Supported for WS-SecureConversation](#) and make a copy.
4. Edit the clone of the policy.
5. Select the **Assertions** tab.
6. Scroll down to the Secure Conversation section of the page.

For the preconfigured WS-SecureConversation policies, secure conversation is enabled by default. For all of the other policies, secure conversation is disabled by default.

7. If you want to simply enable secure conversation for this policy without changing any of the default settings, click **Enabled** and then click **Save** to complete your changes.
8. The following additional controls are available. The default settings depend on the policy type.
 - [Version](#). OWSM supports both Secure Conversation versions 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
 - [Re-authenticate](#)
 - [Client Entropy](#)

- [Derived Keys](#)
 - [Server Entropy](#)
9. The following [Bootstrap Message Security](#) options are available
 - **Inherit From Application Setting.** This is the default: the message security settings for the inner policy are derived from the outer policy.
 - **Use Independent Setting.**
When selected, the following options are available: Algorithm Suite, Encrypt Signature, Confirm Signature, and Include Timestamp.
 10. Click **Configuration**.
The Configuration page displays a list of the configuration properties for the policy.
The `sc.token.lifetime` property specifies the default number of milliseconds after which the secure conversation session should expire. The security context is shared by the client and web service for the lifetime of a communication session. This is the time after which the SCT is expired.
If you do not explicitly set this value, the domain-wide setting applies, as described in [Secure Conversation Configuration for the Domain Using Fusion Middleware Control](#).
 11. If this policy is not already attached to your web service, attach it as described in [Attaching Policies to Manage and Secure Web Services](#).

12.2.2 Configuring Secure Conversation Using WLST

You can configure WS-SecureConversation using WSLT.

To configure WS-SecureConversation:

1. Connect to the running instance of WebLogic Server, as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Begin a session using the `beginWSMSession` command, as described in [Attaching Policies Directly Using WLST](#). For example:

```
wls:/wls_domain/serverConfig> beginWSMSession()
```

```
Session started for modification.
```

3. Select the policy subject that you want to work with. See [Identifying and Selecting the Policy Subject Using WLST](#).
4. Use the `attachWSMPolicy` command to attach a secure conversation policy to a web service port, as described in [Attaching Policies Directly Using WLST](#). For example:

```
wls:/wls_domain/serverConfig> attachWSMPolicy('oracle/  
wss_username_token_over_ssl_wssc_client_policy')
```

```
Policy reference "oracle/wss_username_token_over_ssl_wssc_client_policy" added.
```

5. Use the `setWSMPolicyOverride` command to override policy properties, as described in [Overriding Configuration Properties for Directly Attached Service Policies Using WLST](#).

```
setWSMPolicyOverride(policyURI, property, value)
```

6. Write the contents of the current session to the repository using the `commitWSMSession()` command.

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

12.3 Attaching a Secure Conversation Policy at Design Time

You can attach a WS-SecureConversation policy at design time by using the annotations API.

The API is described in [Understanding Attaching Policies to Web Services and Clients at Design Time](#).

The following examples attach the `oracle/wss11_username_token_with_message_protection_wssc_service_policy` policy alone and in a policy set.

```
@PortableWebService
@oracle.webservices.annotations.SecurityPolicy(
"oracle/wss11_username_token_with_message_protection_wssc_service_policy")

public class TestService {
    ..... }

@PortableWebService
@PolicySet(references=
{@PolicyReference(value="oracle/
wss11_username_token_with_message_protection_wssc_service_policy"),
 @PolicyReference(value="oracle/wsrml1_policy")}
)

public class TestService {
    ..... }
```

12.4 About Configuring Persistence

Each client and web service can specify one or more (one per port) persistence providers, which can be either the Coherence provider or the in-memory provider.

For more information, see "Persistence" in *Understanding Oracle Web Services Manager*.

This section contains the following topics:

- [Overview of Persistence](#)
- [Configuring Persistence for a Web Service](#)
- [Configuring Persistence for a Client](#)

12.4.1 Overview of Persistence

The Coherence persistence provider is the default when running in WebLogic Server, for both the web service client and web service. Otherwise, the in-memory persistence provider is the default.

 **Note:**

You can configure persistence as described in this section only for Oracle Infrastructure web services and clients.

For WebLogic (Java EE) web services, you configure persistence as described in *Configuring Web Service Persistence in Programming Advanced Features of JAX-WS Web Services for Oracle WebLogic Server*.

The following configuration rules apply:

- By default, the client uses the Coherence provider if it is running in WebLogic Server; otherwise, the client uses the in-memory provider. The web service uses the Coherence provider if deployed to WebLogic Server; otherwise, it uses the in-memory provider.
- You can specify one persistence mechanism (per port) for the web service and one for the client, as dictated by your quality of service requirements. There is no requirement that the client and web service use the same `providerName` type, as their configurations may not be the same.
- No matter which mechanism you choose, if `providerName` is blank, `oracle:jrf:Coherence` is used if available; otherwise `oracle:jrf:Memory` is used.
- If you explicitly set `oracle:jrf:Coherence` and it is not available, an error is returned.

To configure persistence, you can use any of the following mechanisms. In each case, you specify only the `providerName`, which can be only `oracle:jrf:Memory` or `oracle:jrf:Coherence`.

12.4.2 Configuring Persistence for a Web Service

This topic describes how to configure persistence for a Web Service.

- Use the `@Persistence` policy annotation in the web service at design time, as described in [@Persistence](#) and shown in the following example:

```
@PortableWebService
@SecurityPolicy("oracle/
wss11_username_token_with_message_protection_wssc_service_policy")
@Persistence(providerName="oracle:jrf:Coherence")

public class TestService {
    .....
}
```

- Use the `@PolicyReference` policy annotation in the web service at design time, as described in [Understanding Attaching Policies to Web Services and Clients at Design Time](#) and shown in the following example:

```
@PortableWebService
@SecurityPolicy("oracle/
wss11_username_token_with_message_protection_wssc_service_policy")
@PolicySet(references={
    @PolicyReference(value="oracle/
wss11_username_token_with_message_protection_wssc_service_policy"),
    @PolicyReference(value="oracle/persistence_policy")
})

public class TestService {
```

```
.....
}
```

- Attach the `oracle/persistence_policy` policy and set the `providerName` attribute via Fusion Middleware Control or WLST post deployment, as described in [About Attaching Policies to Web Services and Clients Using Fusion Middleware Control](#) and [About Attaching Policies to Web Services and Clients Using WLST](#) respectively.

12.4.3 Configuring Persistence for a Client

This topic describes how to configure persistence for a Client.

- Use the `policy-reference` element in the client at design time and set the `provider.name` property, as shown in the following example:

```
String configString =
"<port-info>\n" +
" <policy-references>\n" +
" <policy-reference uri=\"oracle/
wss11_username_token_with_message_protection_wssc_client_policy\"
category=\"security\"/>\n" +
" <policy-reference uri=\"oracle/persistence_policy\" category=\"wsconfig\">"
+
" <property name=\"provider.name\" value=\"oracle:jrf:Memory\"/>" +
" </policy-reference>" + " </policy-references>\n" + "</port-info>";
// convert configString to Element

((BindingProvider) port).getRequestContext().put(
    ClientConstants.CLIENT_CONFIG, configElement);
```

- Set `PersistenceFeature` at design time, as shown in the following example:

```
PersistenceFeature persistenceFeature =
    PersistenceFeature.builder().providerName("oracle:jrf:Memory").build();

port = service.getPort(new WebServiceFeature[] {persistenceFeature});
```

- Attach the `oracle/persistence_policy` policy and set the `providerName` attribute via Fusion Middleware Control or WLST post deployment, as described in [About Attaching Policies to Web Services and Clients Using Fusion Middleware Control](#) and [About Attaching Policies to Web Services and Clients Using WLST](#) respectively.

12.5 Understanding Secure Conversation Sessions

OWSM maintains the client and server secure conversation session information based on a computed Session ID. OWSM (via an internal session mechanism) computes the Session ID at runtime for each message, and associates one or more requests to a session.

This is described in "WS-SecureConversation Architecture".

The session ID is generally reused if:

- A subsequent request comes from the same client and the session ID has not already been freed. (If the session ID has already been freed, a new session ID is created.)
- The `sc.token.lifetime` property has not expired.

Once created, Secure Conversation session ID's are not removed automatically until the underlying Java virtual machine used to start the WebLogic Server instance shuts down.

The session management WLST commands provide a way for you to free up the session ID's on the server in the following situations:

- You detect any issue with server performance and you want to free up the resources on the server.
- The SCT token has expired. There is an expiration time displayed by the `getServiceSessionInfo()` command. If the SCT token has expired, you can remove the session ID.

 **Note:**

The scope of session is the current Persistence provider. That is, you can list and remove only those session ID's that are stored within the current Persistence provider.

OWSM provides the following WLST commands that you can use to manage Secure Conversation sessions.

 **Note:**

All of the WebLogic Server instances within a domain must be running in order for the WLST commands to succeed.

- `listWebServiceSessionNames()` — Lists the names of all sessions visible within the instance of a running server. The returned names are appropriate for use as the name parameter in subsequent calls to `getServiceSessionInfo(String)` and `removeWebServiceSession(String)` commands.

For example:

```
wls:/base_domain/serverConfig> listWebServiceSessionNames ()
215d0d4a5ebbc3fec662f46adedc5bc74ecbc87b
```

- `getServiceSessionInfo()` — Given a session name, get a data object representing current information for that session. The returned information is as follows.
 - name
 - creation time
 - last update time
 - expiration time
 - `KeyInfo[]` user-defined keys, where `KeyInfo` is `keyName` and `keyValue`.

For example:

```
wls:/base_domain/serverConfig>
getServiceSessionInfo ('215d0d4a5ebbc3fec662f46adedc5bc74ecbc87b')
Name: 215d0d4a5ebbc3fec662f46adedc5bc74ecbc87b
Creation time: Mon Nov 04 17:47:39 PST 2013
Last update time: Mon Nov 04 17:47:42 PST 2013
Expiration time: Mon Nov 04 18:17:41 PST 2013
Key info: [oracle.wsm.security.seconv.util.property.SCT,
```

```
0x0000014225F1A1260AE4F30351FD1544DC10ED14201988C8CFEDFDBE8E0E4B09  
]
```

- `listWebServiceSessionNamesForKey()` — Lists the names of all sessions identified by `keyName` and `keyValue`.

The returned names are appropriate for use as the name parameter in subsequent calls to `getWebServiceSessionInfo(String)` and `removeWebServiceSession(String)` commands.

For example:

```
wls:/base_domain/serverConfig>  
listWebServiceSessionNamesForKey('oracle.wsm.security.seconv.util.property.SC  
T',  
'0x0000014225F1A1260AE4F30351FD1544DC10ED14201988C8CFEDFDBE8E0E4B09')  
215d0d4a5ebbc3fec662f46adedc5bc74ecbc87b
```

- `removeWebServiceSession()` — Given a session name, removes the session object associated with that session name.

For example:

```
wls:/base_domain/serverConfig>  
removeWebServiceSession('215d0d4a5ebbc3fec662f46adedc5bc74ecbc87b')
```

For more information about the WLST commands and their arguments, see "Session Management WLST Commands" in *WLST Command Reference for WebLogic Server*.

13

Integrating Hardware with Oracle Web Services Manager

Hardware, such as SafeNet Luna SA, can be integrated with OWSM to provide a secure way to store keys and off-load cryptographic processing. Integrating OWSM with Oracle SPARC T5 and SPARC T4 Cryptographic Acceleration allows you to use the T5 and T4 processor based cryptographic capabilities.

Integrating OWSM with required hardware 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 chapter includes the following sections:

- [Using Hardware Security Modules With OWSM](#)
- [About Configuring OWSM for Oracle SPARC T5 and SPARC T4 Cryptographic Acceleration](#)

13.1 Using Hardware Security Modules With OWSM

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. OWSM supports the SafeNet Luna SA HSM, which you can install and configure.

The following sections explain this further:

- [Understanding SafeNet Luna SA With OWSM for Key Storage](#)
- [About Installing and Configuring the Luna SA HSM Client](#)
- [Configuring the JRE Used By OWSM](#)
- [Logging On to Luna SA](#)
- [Copying Keys and Certificates to Luna SA](#)
- [About Configuring OWSM to Use Luna SA](#)

13.1.1 Understanding SafeNet Luna SA With OWSM 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 OWSM 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, OWSM uses the Keystore Service (KSS) for key storage. Keys and certificates required by OWSM for cryptographic operations are fetched from a keystore file. When Luna SA is available in-network, it can be leveraged by OWSM for key storage purposes and cryptographic operations.

13.1.2 About Installing and Configuring the Luna SA HSM Client

You need to install the Luna SA HSM client on the host that has a running instance of OWSM so that it can 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 installing the Luna SA HSM client, verify the following checklist:

- You already have Luna SA installed and available in your 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 obtain the hardware security module, and to acquire the necessary library.

These tasks must be performed before you can use a Luna SA hardware security module with OWSM.

13.1.3 Configuring the JRE Used By OWSM

After installing the Luna SA client, you need to configure the JRE that will be used by the OWSM setup.

You must perform the following steps:

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.

To do so, add these two Luna providers at the end of the `security.providers` list:

```
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.

13.1.4 Logging On to Luna SA

Before you can use Luna SA with OWSM, you must log on to the Luna SA server. This is an one-time process that creates a Luna log-in session on the client machine and the 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>
```

13.1.5 Copying Keys and Certificates to Luna SA

You need to move all keys and certificates to Luna SA, if the keys and certificates are currently in a KSS or JKS keystore. 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.

**Note:**

The `cmu` script imports from a file. Therefore, before you can import the keys and certificates into Luna SA, you must first export them to a file from a KSS or JKS keystore.

13.1.6 About Configuring OWSM to Use Luna SA

As part of configuring OWSM to use Luna SA, you need to configure the OWSM domain to use the Luna HSM instead of the default KSS keystore.

For more information, see the following topics:

- [Configuring OWSM to Use HSM Keystores](#)
- [Configuring the OWSM Keystore Using WLST](#)

13.2 About Configuring OWSM for Oracle SPARC T5 and SPARC T4 Cryptographic Acceleration

You can configure OWSM to take advantage of cryptographic acceleration capabilities of Oracle SPARC T5 and SPARC T4 processor-based servers.

OWSM supports the use of Oracle SPARC T5 and 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 OWSM on Oracle SPARC T5 and SPARC T4 based servers transparently leverages the SPARC T5 and 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 applies only to users who are running Oracle SPARC T5 and T4 processor-based servers running Oracle Solaris 10 8/11 or later.

The following topics are described:

- [Terms You Need to Understand](#)
- [Overview of Oracle SPARC T5 and SPARC T4 Hardware Assisted Cryptographic Acceleration](#)
- [Configuring Transport-Level Security for Cryptographic Acceleration](#)
- [Configuring Message-level Security for Cryptographic Acceleration](#)
- [Additional Reading for Cryptographic Acceleration](#)

13.2.1 Terms You Need to Understand

There are certain terms that you need to understand to configure OWSM for cryptographic acceleration.

Refer to the white paper described in [Additional Reading for Cryptographic Acceleration](#) 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.

13.2.2 Overview of Oracle SPARC T5 and SPARC T4 Hardware Assisted Cryptographic Acceleration

The Oracle SPARC T5 and SPARC T4 processors are 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 T5 and SPARC T4 processors 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 OWSM on Oracle SPARC T5 and SPARC T4 based servers delivers high performance by leveraging the 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.

OWSM makes use of SPARC T5 and SPARC T4 processor based cryptographic acceleration in the following scenarios:

- Transport-level security, as described in [Configuring Transport-Level Security for Cryptographic Acceleration](#).

The SPARC T5 and SPARC T4 processor on-core cryptographic acceleration capabilities can be accessed in a variety of ways by the OWSM deployment, depending on the applied security scenarios and its requirements. The availability of Oracle Ucrypto provider features and the Sun PKCS#11 interfaces in the Java Cryptography Extension (JCE) framework enable OWSM and WebLogic Server to take advantage of hardware-assisted cryptographic acceleration of SSLv3/TLSv1 and WS-Security-based cryptographic workloads.

With the release of JDK7 update 4, Oracle introduced Oracle Ucrypto provider, which provides a specialized interface that bypasses PKCS#11 and automatically leverages hardware-assisted cryptographic acceleration capabilities of Oracle's SPARC T4 and SPARC T5 (or newer) processors. In a typical JDK7 installation (JDK7u4 or later) on Oracle Solaris 11 (SPARC), the Java Runtime Environment is preconfigured to make use of the Oracle Ucrypto provider by default. This enables the Java and Oracle WebLogic Server-hosted applications and XML web services to automatically delegate their cryptographic-intensive operations processed via Oracle Solaris Cryptographic Framework using Oracle SPARC T5- and SPARC T4-based on-core cryptographic instructions.

In JDK6, the Java PKCS#11 interfaces help to off-load and accelerate the compute-intensive cryptographic workloads of SSL/TLS protocols by leveraging the on-core cryptographic instructions of SPARC T5 and SPARC T4 processors.

- Message-level security, as described in [Configuring Message-level Security for Cryptographic Acceleration](#).

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.

OWSM implements a dedicated PKCS#11 interface to delegate cryptographic operations (via SCF) to on-core cryptographic instructions of SPARC T4 processor.

13.2.3 Configuring Transport-Level Security for Cryptographic Acceleration

You can configure cryptographic acceleration for transport-level security.

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 Online 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 "Using the JSSE-Based SSL Implementation" in *Administering Security for Oracle WebLogic Server 14c (14.1.2)* for the steps to follow.

3. In a typical WebLogic Server installation on SPARC Solaris, the Java runtime environment is pre-configured to make use of the Oracle Ucrypto Provider (JDK7) or SunPKCS11 provider.

To leverage the Oracle Ucrypto provider, you must use JDK7 update 4 (or later) as the Java Runtime Environment on Oracle Solaris 11. Make sure that the Oracle Ucrypto provider is identified as the default provider in the Java security properties file `java.security` located in the `$JAVA_HOME/jre/lib/security/` directory.

```
security.provider.1=com.oracle.security.ucrypto.UcryptoProvider
${java.home}/lib/security/ucrypto-solaris.cfg
```

4. If using JDK6, 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 the following example of a 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
```

5. Restart WebLogic Server.
6. Verify that the SSL configuration is working.

13.2.4 Configuring Message-level Security for Cryptographic Acceleration

You can configure cryptographic acceleration for message-level security.

To configure the 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 softtoken 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.cfg` `sunpkcs11-solaris.cfg` configuration file.

For example, if the specified algorithm suite for a policy is Basic256Rsa15 as shown in [Figure 13-1](#), 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 13-1 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 OWSM keystore to use the PKCS11 keystore as described in the following sections:
 - [Configuring OWSM to Use the PKCS11 Keystore](#)
 - [Configuring the OWSM Keystore Using WLST](#)
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.

13.2.5 Additional Reading for Cryptographic Acceleration

Certain whitepapers are a definitive source for cryptographic acceleration information for using Oracle SPARC T-series processor based servers. These whitepapers cover many additional pertinent topics such as Solaris Cryptographic Framework components, using Solaris Kernel SSL (KSSL), Apache Web Server, Oracle Database Transparent Data Encryption and performance characteristics.

Refer to the following whitepapers for more information:

- For information on using high-performance security on SPARC T5 and SPARC M5 servers, refer to the whitepaper *"High-Performance Security for Oracle WebLogic Server Applications Using Oracle's SPARC T5 and SPARC M5 Servers"*, which is available at

<http://www.oracle.com/technetwork/articles/systems-hardware-architecture/security-weblogic-t-series-168447.pdf>.

- For information on OWSM deployments on Oracle SPARC T-series processor based servers, refer to the whitepaper "*High Performance Security For Oracle Database and Fusion Middleware Applications using SPARC T4*", which is available at <http://www.oracle.com/technetwork/server-storage/sun-sparc-enterprise/documentation/o12-021-t4security-1577047.pdf>.

Part IV

Managing and Troubleshooting Oracle Web Services Manager

Part IV describes how to manage the configuration of Oracle Web Services Manager (OWSM) at the domain level, how to manage the OWSM Repository, and how to troubleshoot problems with OWSM.

It contains the following chapters:

- [Managing Oracle Web Services Manager Domain Configuration](#), describes how to manage the configuration of OWSM at the domain level for authentication, message protection, and policy access.
- [Managing the Oracle Web Services Manager Repository](#), describes how to work with OWSM Repository. You can register, backup, restore, and rebuild the repository. You can also import, export, and upgrade policies in the repository. import and export, upgrade documents,
- [Diagnosing Problems with Oracle Web Services Manager](#), describes how to use Fusion Middleware Control and WLST to troubleshoot problems with Oracle Policy Manager and policy attachment issues.
- [Managing third party server integration with OWSM](#), describes how to manage OWSM integration with 3rd party servers like Google OAUTH2 server and Twitter OAuthServer.

Managing Oracle Web Services Manager Domain Configuration

This chapter describes how to configure the OWSM environment for authentication, message protection, and policy access for the domain. You can perform this configuration using either Fusion Middleware Control or WLST. Both methods are described.

This chapter includes the following sections:

- [Overview of OWSM Domain Configuration](#)
- [Navigating to the WSM Domain Configuration Page](#)
- [Viewing the General OWSM Domain Configuration Using Fusion Middleware Control](#)
- [Configuring Domain-Level Authentication Using Fusion Middleware Control](#)
- [Domain-Level Message Security Configuration Using Fusion Middleware Control](#)
- [OWSM Policy Access Configuration Using Fusion Middleware Control](#)
- [About Managing OWSM Domain Configuration Properties Using WLST](#)
- [Configuring Domain-Level Authentication Using WLST](#)
- [About Configuring Domain-Level Message Security Using WLST](#)
- [About Configuring Policy Access Using WLST](#)

14.1 Overview of OWSM Domain Configuration

OWSM provides the capability for an administrator to configure and manage the OWSM environment from a central location using the WSM Domain Configuration page in Fusion Middleware Control. From this page you can specify the Policy Manager connection information, cache refresh rates, nonce timeout, clock skew, OWSM keystore configuration, login modules, trusted SAML issuers, and other similar configuration parameters.

WLST commands also provide the ability to configure OWSM from the command line or using scripts. For more information about the WLST commands, see "[About Managing OWSM Domain Configuration Properties Using WLST](#)".

OWSM configurations are stored as documents in the OWSM Repository. When you install Fusion Middleware and create a domain that includes OWSM, the following configuration documents are created by default:

- Bootstrap configuration document—Contains bootstrap information that is used to connect to the Policy Manager. If you modify the domain configuration (using Fusion Middleware Control or WLST) for a property that is also specified in the bootstrap configuration document, then the value specified in the bootstrap configuration is used only for the initial Policy Manager connection. For any subsequent access, the value in the domain configuration document is used.

The bootstrap configuration document is stored in the `domain_home/config/fmwconfig/wsm-config.xml` file, where `domain_home` is the directory in which you installed your domain. By default this directory is `Oracle_Home/user_projects/domains/base_domain`.

- Token issuer trust document—Contains the default trusted SAML issuer, `www.oracle.com`. This document is read-only. As you add trusted issuers and define DN lists, additional trust documents are created and stored in the repository.
- Default configuration document—Contains all of the properties valid for the context with their default values. When you modify the default configuration, a configuration document for the domain is created and saved in the repository.

Note that the configuration performed using the WSM Domain Configuration page or using the custom configuration WLST commands applies at the domain level only, not to the application level.

In most cases, changes made to the OWSM configuration do not require a server restart. In certain situations, however, such as changing the keystore configuration or keystore type, changes to the configuration require a server restart. Additionally, if you modify any cache properties (such as `cache.refresh`) and you want the changes to go into effect immediately, you should restart the server.

14.2 Navigating to the WSM Domain Configuration Page

You manage the configuration of OWSM using the WSM Domain Configuration page. From this page you can view general information about the domain, and configure authentication, message security, and policy access properties. This topic describes the procedure to navigate to the WSM Domain Configuration page.

To navigate to the WSM Domain Configuration page:

1. From the navigation pane, expand **WebLogic Domain** and select the domain to be configured.
2. From the **WebLogic Domain** menu, select **Web Services**, then **WSM Domain Configuration**.

14.3 Viewing the General OWSM Domain Configuration Using Fusion Middleware Control

The **General** tab on the WSM Domain Configuration page provides basic information about the domain such as domain name and platform type, and enables you to provide a display name and description for the domain. This section describes the procedure to view this general information.

Version information about the domain is also shown, including the version number of the configuration, timestamp for the last update to the configuration, and the name of the user who last updated the domain.



Note:

Version information is updated only if you use a database-based OWSM Repository.

To view general information about the domain:

1. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".

2. Select the **General** tab.
3. View the basic information about the domain, including **Name**, **Platform Type**, and **Version Information**.
4. Optionally, provide a **Display Name** and **Description** for the configuration.
5. Click **Apply**.

14.4 Configuring Domain-Level Authentication Using Fusion Middleware Control

The **Authentication** tab on the WSM Domain Configuration page provides the ability to configure SAML trust, the lifetime to be used for the issued token, and subject properties for JAAS subjects that are created after OWSM authentication. You can also configure SAML, SAML2, Kerberos, X509, and custom login modules.

The configuration details are described in the following sections:

- [SAML Trusted Issuers and DN Lists Using Fusion Middleware Control](#)
- [Configuring JWT Trusted Issuers and DN Lists Using Fusion Middleware Control](#)
- [Configuring Token Attribute Rules for Trusted Issuers Using Fusion Middleware Control](#)
- [Configuring the Lifetime for the Issued Token Using Fusion Middleware Control](#)
- [Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)
- [Configuring the Kerberos Login Module](#)
- [Configuring Subject Properties Using Fusion Middleware Control](#)
- [Configuring the X509 Login Module Using Fusion Middleware Control](#)
- [Creating Custom Login Modules](#)

14.4.1 SAML Trusted Issuers and DN Lists Using Fusion Middleware Control

The SAML Trust section on the Authentication tab enables you to define SAML trusted issuers and a list of trusted distinguished names (DNs) for SAML signing certificates.

The following topics provide information to help you configure SAML trusted issuers and DN lists:

- [Overview of SAML Trusted Issuers and DN Lists](#)
- [Adding SAML Issuers and Defining a Trusted DN List Using Fusion Middleware Control](#)
- [Deleting Trusted Issuers, DN, or DN Lists Using Fusion Middleware Control](#)

To complete these tasks using WLST, see "[Configuring SAML and JWT Trusted Issuers, DN Lists, and Token Attribute Rules Using WLST](#)".

14.4.1.1 Overview of SAML Trusted Issuers and DN Lists

The list of SAML issuers that you define using the Authentication page 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.

By default, OWSM checks the incoming issuer name against the list of configured issuers, and checks the SAML signature against the configured certificates in the OWSM keystore. If you defined a trusted DNS list for a trusted issuer, OWSM also verifies that the SAML signature is signed by a certificate whose DN belongs to the trusted DN list.

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 DNSs for a trusted issuer, then OWSM allows signing by any certificate, as long as all the intermediate certificates and the CA certificate in the certificate chain for the signing certificate are present in the OWSM keystore. If the signing certificate is self-signed, it must be in the keystore itself.

Important Notes:

- Using the Trusted STS and Trusted Clients tables in the SAML Trust section, you define the DNSs of the *signing certificates*, not the certificates themselves.
- The CA and intermediate certificates for the signing certificate must be in the OWSM keystore regardless of whether the signing certificate is in the keystore or passed in the message.
- For two-way SSL:
 - The certificate needs to be imported into the trust store for the Java EE container.
 - The DN of the client SSL certificates are used for validation and must be present in the trusted DNSs list.
- In all cases, the signing certificates must be trusted by the certificates present in the OWSM keystore.

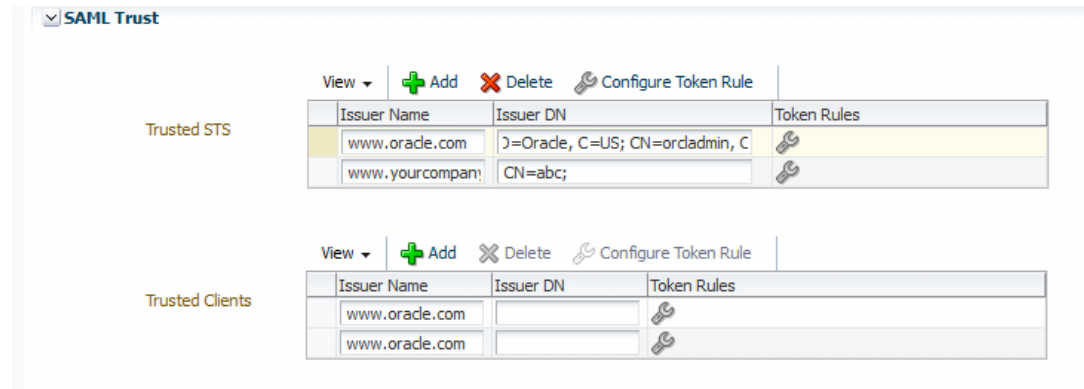
14.4.1.2 Adding SAML Issuers and Defining a Trusted DN List Using Fusion Middleware Control

Use the following procedure to add SAML issuers, define a trusted DN list for SAML signing certificates, or to add a DN to the DN list for an issuer:

1. Access the WSM Domain Configuration page, as described in [Navigating to the WSM Domain Configuration Page](#)
2. Select the **Authentication** tab.
3. In the SAML Trust section of the page, shown in [Figure 14-1](#), do one of the following:
 - To add a trusted issuer and define a trusted DN list for trusted STS servers, click **Add** in the Trusted STS table. Use this list for SAML HOK and SAML bearer.
 - To add a trusted issuer and define a trusted DN list for trusted clients, click **Add** in the Trusted Clients table. Use this list for SAML sender vouches.
4. In the **Issuer Name** column, enter a trusted issuer, for example `www.example.com`. The default value for the predefined SAML client policies is `www.oracle.com`.
5. In the **Issuer DN** column, enter the DN list for the trusted issuer. Use a string that conforms to RFC 2253. For example, the trusted DN for the trusted issuer `www.oracle.com` is `CN=weblogic, OU=Oracle Test Encryption Purposes Only, O=Oracle, C=US`. To add more than one DN for an issuer, separate each DN with a semicolon (;), as shown in [Figure 14-1](#).

For more information about RFC 2253, see <http://www.ietf.org/rfc/rfc2253.txt>.

Figure 14-1 SAML Trust Section of Authentication Tab



6. Optionally, configure token attribute rules for the trusted DN, as described in "[Configuring Token Attribute Rules for Trusted Issuers Using Fusion Middleware Control](#)".
7. Optionally, add additional trusted issuers and DN lists. To do so, repeat steps 3 through 6.
8. To add a DN to the DN list for a trusted issuer, select the trusted issuer in the table and append the DN to the list of DNs in the **Issuer DN** column. Be sure to separate each DN by a semicolon (;).
9. When you have finished configuring the necessary issuers and DN lists, click **Apply**.

14.4.1.3 Deleting Trusted Issuers, DNs, or DN Lists Using Fusion Middleware Control

You can delete issuers, a DN, or a DN list by performing one of the following actions:

- To delete a trusted issuer and the DN list, select the row containing the issuer to be deleted and click **Delete**.
- To delete a DN from the DN list, select the row for the trusted issuer and delete the DN from the list.
- To delete a DN list, clear the **Issuer DN** field for the DN to be deleted. Note that any configured token rules are also deleted.

14.4.2 Configuring JWT Trusted Issuers and DN Lists Using Fusion Middleware Control

The JWT Trust section on the Authentication tab enables you to define JWT trusted issuers and a list of trusted distinguished names (DNs) for JWT signing certificates.

The following topics provide information to help you configure JWT trusted issuers and DN lists:

- [About JWT Trusted Issuers and DN Lists](#)
- [Adding JWT Issuers and Defining a Trusted DN List Using Fusion Middleware Control](#)

To delete issuers, a DN, or a DN list, "[Deleting Trusted Issuers, DNs, or DN Lists Using Fusion Middleware Control](#)".

To complete these tasks using WLST, see "[Configuring SAML and JWT Trusted Issuers, DN Lists, and Token Attribute Rules Using WLST](#)".

14.4.2.1 About JWT Trusted Issuers and DN Lists

The list of JWT issuers that you define using the Authentication page 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.

By default, OWSM checks the incoming issuer name against the list of configured issuers, and checks the JWT signature against the configured certificates in the OWSM keystore. If you defined a trusted DN list for a trusted issuer, OWSM also verifies that the JWT signature is signed by a certificate whose DN belongs to the trusted DN list.

Configuration of the trusted DN 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 DN for a trusted issuer, then OWSM allows signing by any certificate, as long as all the intermediate certificates and the CA certificate in the certificate chain for the signing certificate are present in the OWSM keystore. If the signing certificate is self-signed, it must be in the keystore itself.

14.4.2.2 Adding JWT Issuers and Defining a Trusted DN List Using Fusion Middleware Control

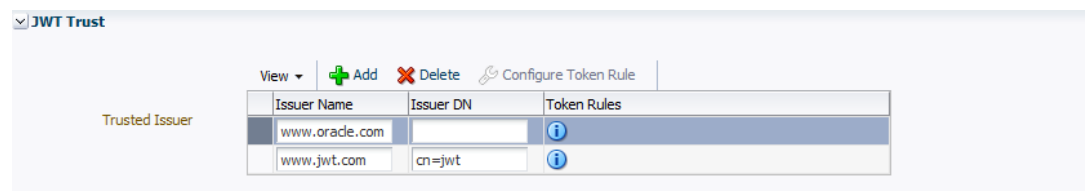
This section describes the procedure to add JWT issuers, define a trusted DN list for JWT signing certificates, or to add a DN to the DN list for an issuer.

To add JWT issuers, define a trusted DN list:

1. Access the WSM Domain Configuration page, as described in [Navigating to the WSM Domain Configuration Page](#).
2. Select the **Authentication** tab.
3. In the JWT Trust section of the page, shown in [Figure 14-1](#), click **Add** in the Trusted Issuer table, to add a trusted issuer and define a trusted DN list.
4. In the **Issuer Name** column, enter a trusted issuer, for example `www.example.com`. The default value for the predefined JWT client policies is `www.oracle.com`.
5. In the **Issuer DN** column, enter the DN list for the trusted issuer. Use a string that conforms to RFC 2253. For example, the trusted DN for the trusted issuer `www.oracle.com` is `CN=weblogic, OU=Oracle Test Encryption Purposes Only, O=Oracle, C=US`. To add more than one DN for an issuer, separate each DN with a semicolon (;), as shown in [Figure 14-1](#).

For more information about RFC 2253, see <http://www.ietf.org/rfc/rfc2253.txt>.

Figure 14-2 JWT Trust Section of Authentication Tab



6. Optionally, configure token attribute rules for the trusted DN, as described in "[Configuring Token Attribute Rules for Trusted Issuers Using Fusion Middleware Control](#)".

7. Optionally, add additional trusted issuers and DN lists. To do so, repeat steps 3 through 6.
8. To add a DN to the DN list for a trusted issuer, select the trusted issuer in the table and append the DN to the list of DNs in the **Issuer DN** column. Be sure to separate each DN by a semicolon (;).
9. When you have finished configuring the necessary issuers and DN lists, click **Apply**.

14.4.3 Configuring Token Attribute Rules for Trusted Issuers Using Fusion Middleware Control

There are increasing requirements to control which users and user attributes are accepted and processed for a particular trusted SAML or JWT token issuer. Token attribute rules allow you to define additional security constraints for the trusted STS server and for the trusted SAML client. Token attribute rules can be defined on top of a trusted DN. 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 user attributes that a DN for a trusted issuer can assert. The name ID and each attribute in the attributes part can contain a filter with multiple values or value patterns to provide constraints for the value of the name ID or the attribute that the DN can assert.

To define token attribute rules using Fusion Middleware Control:

1. Define a DN list for a trusted issuer as described in "[SAML Trusted Issuers and DN Lists Using Fusion Middleware Control](#)".
2. Select the Issuer DN in the table and click **Configure Token Rule**.
3. In the Token Rule page, click **Add** to add attributes and filters for the DN.

The **Attribute** field is prepopulated with the value `name-id`.

 **Note:**

Only the first row is prepopulated with the value `name-id` in the **Attribute** field. When you click **Add** for subsequent rows, you need to enter a value in the **Attribute** field.

4. In the **Filters** field, enter the attribute filter values to create the token attribute rule.
The value that you add in the **Filter** field can be a complete name, such as `yourTrustedUser`. You can also enter a name pattern with a wildcard character (*), such as `yourTrusted*`. If you specify multiple attribute filters, each filter should be separated by a semicolon (;).
5. To edit an existing attribute rule or filter, select the appropriate field in the table and make the required changes.
6. To delete an attribute rule, select the row containing the rule to be deleted and click **Delete**.
7. Click **OK** to save your changes and return to the WSM Domain Configuration page.
8. When you have finished configuring the necessary changes, click **Apply**.

14.4.4 Configuring the Lifetime for the Issued Token Using Fusion Middleware Control

You can specify the lifetime to be used for the issued token when the request message is sent to the Security Token Service (STS).

If the STS sends the token with a different expiry, the runtime looks at the actual expiry of the token and only caches the token for that time. For more information, see "[Understanding Token Lifetime and Token Caching](#)".

The default for this property is 28800000 milliseconds (8 hours). To modify the default value, do one of the following:

- Enter the desired value directly into the **Issued Token Lifetime** field.
- Use the up/down arrows to increase or decrease the default value.

14.4.5 Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control

You can configure the SAML and SAML2 login modules from the OWSM Domain Configuration page.

The SAML 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 module and make any needed changes.

You can configure the following SAML login modules from the OWSM Domain Configuration page:

- `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.

 **Note:**

The configuration performed on this page takes precedence over any configuration performed in the OPSS login module page.

To configure the SAML login modules in OWSM:

1. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
2. Select the **Authentication** tab.
3. In the SAML or SAML2 Login Module section of the page, set the following properties as appropriate for your environment:

- **Allow Virtual User**—Select this property to allow the SAML subject to be treated as a virtual user. If enabled, 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.

If this property is not enabled (the default), 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.

- **Domain Name Mapping Attribute**—If the Name Identifier Format is set to X509SubjectName in the SAML client policy, you can specify which part of the certificate DN to use for asserting the user against the identity store. The default for this property is CN.
- **Add Assertion To Subject**—Specify if the SAML assertion should be added to the authenticated subject as a private credential. The default is true. If you set this property to false (clear the checkbox), then the assertion is not added to the authenticated subject.

4. Click **Apply**.

For details about setting these properties using WLST, see "[Configuring the SAML and SAML2 Login Modules Using WLST](#)".

14.4.6 Configuring the Kerberos Login Module

The Kerberos policies have an associated JAAS login module that authenticates users using Kerberos protocols. The Kerberos login module has optional properties that you can configure in OWSM. The configuration specified in the OWSM configuration takes precedence over any Kerberos login module configuration in OPSS.

To configure the Kerberos login module in OWSM:

1. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
2. Select the **Authentication** tab.
3. In the Kerberos section of the page, set the following properties as appropriate for your environment:
 - **Keytab file location**—Specify the file name of the keytab to get the secret key for the principal. The default value is `./krb5.keytab`.
 - **Principal**—Specify the name of the principal to be used. The principal represents a specific entity to which a set of credentials are assigned. It can be a simple username, such as `testuser`, or a service name such as `HOST/localhost`. 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. The default value is `HOST/localhost@EXAMPLE.COM`.
 - **Class Name**—Specify the login module to be used by OWSM. If no value is specified, at runtime one of the following default values will be used for the Kerberos login module Class Name, depending on the platform on which you are running:
 - `com.sun.security.auth.module.Krb5LoginModule` (Oracle)
 - `com.ibm.security.auth.module.Krb5LoginModule` (IBM)

- **Use Keytab**—Specifies whether the module will get the principal's key from the keytab. If the 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`.

Select the checkbox to set to `true`, that is, to get the principal's key from the keytab.

- **Store Key**—Specifies whether the principal's key is stored in the Subject's private credentials.

Select the checkbox to set to `true`, that is, to store the principal's key in the Subject's private credentials.

- **Do Not Prompt for Credentials**—Specifies whether you will be prompted for the password if credentials cannot be obtained from the cache or keytab. When this property is selected, authentication will fail if credentials can not be obtained from the cache or keytab.

Select the checkbox to set to `true`, that is, to be prompted for the password if credentials cannot be obtained from the cache or keytab.

4. Click **Apply**.

For details about setting these and other Kerberos login module properties using WLST, see "[Configuring the Kerberos Login Module Using WLST](#)".

14.4.7 Configuring Subject Properties Using Fusion Middleware Control

The JAAS subject that is created after OWSM authentication is populated with usernames and roles, including the authenticated role, anonymous role, and application roles.

For more information about these roles, see "Understanding Users and Roles" in *Securing Applications with Oracle Platform Security Services*.

OPSS supports subject properties for these roles that can be configured in OWSM and passed to OPSS with each invocation. The configuration defined in OWSM applies to all subjects created in OWSM only. It does not affect any subjects created using the OPSS login modules.

To configure subject properties in OWSM:

1. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
2. Select the **Authentication** tab.
3. In the Subject Properties section of the page, set the following properties as appropriate for your environment:

- **Remove Anonymous Role from Subject**—Specify whether the anonymous role is removed from the subject created in OWSM. By default, this property is disabled (set to `false`), indicating that the subject will retain the anonymous role after authentication.

Select the checkbox (set to `true`) to specify that the anonymous role will be removed from the subject after authentication.

- **Authenticated Role to Subject**—Specify whether the authenticated role is included in the subject created in OWSM. By default, this property is enabled (set to `true`) to indicate that the authenticated role is included in the subject.

Clear the checkbox (set to `false`) if the authenticated role is not to be included in the Subject.

- **Application Role to Subject**—Specify whether the application role is added to the subject created in OWSM. By default, this property is enabled (set to true), indicating that the application role is added to the subject.

Clear the checkbox (set to `false`) to specify that the application role is not added to the subject.

4. Click **Apply**.

For details about setting these properties using WLST, see "[Configuring Subject Properties Using WLST](#)".

14.4.8 Configuring the X509 Login Module Using Fusion Middleware Control

The X509 login module has optional properties that you can configure in OWSM. The configuration specified in the OWSM configuration takes precedence over any X509 login module configuration in OPSS.

To configure the X509 login module:

1. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
2. Select the **Authentication** tab.
3. In the X509 section of the page, set the following property as appropriate for your environment:
 - **Domain Name Mapping Attribute**—Enter the desired mapping attribute for a login using DN. The default for this property is CN, but you can specify any part of the certificate DN to use for asserting the user against the identity store. The value that you specify is passed to OPSS with each invocation of the login module. The DN string should conform to RFC 2253.

For more information about RFC 2253, see <http://www.ietf.org/rfc/rfc2253.txt>.

4. Click **Apply**.

14.4.9 Creating Custom Login Modules

You can create custom login modules in the OWSM configuration system using Fusion Middleware Control.

For details about creating a login module, see the *Java Authentication and Authorization Service (JAAS) LoginModule Developer's Guide* at <http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASLMDevGuide.html>.

To create a custom login module:

1. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
2. Select the **Authentication** tab.
3. In the Custom Login Module section of the page, click **Create**.
4. In the Create Login Module page, enter a name and the classname for the login module in the **Name** and **Class** fields, respectively. Optionally provide a description of the login module in the **Description** field.

For example, to configure the Keystore login module, enter `Keystore Login Module` in the **Name** field and `com.sun.security.auth.module.KeyStoreLoginModule` in the **Class** field.

- To add custom properties, click **Add** and provide a name and value for the property. Repeat until you have added all the required properties. To delete a property, select the row and click **Delete**.

For example, if you are configuring the Keystore Login Module, you can add a `keyStoreAlias` property, as shown in [Figure 14-3](#).

Figure 14-3 Custom Login Module Page

The screenshot shows a 'Create Login Module' dialog box. It contains the following fields and controls:

- Name:** Keystore Login Module
- Class:** com.sun.security.auth.module.Key
- Description:** (empty)
- Custom Properties:** A table with two columns: Name and Value. It contains one row with Name 'keyStoreAlias' and Value 'orakey'.
- Buttons:** '+ Add' and 'X Delete' are located above the table. 'Ok' and 'Cancel' are at the bottom right.

- Click **OK** to create the login module.
- Click **Apply**.

14.5 Domain-Level Message Security Configuration Using Fusion Middleware Control

The **Message Security** tab on the WSM Domain Configuration page provides the ability to configure the OWSM keystore, tune security policy enforcement, specify if the public certificate should be published in the WSDL, and configure hostname verification and secure conversation.

The configuration details are described in the following sections:

- [OWSM Keystore Configuration Using Fusion Middleware Control](#)
- [Configuring Security Policy Enforcement Using Fusion Middleware Control](#)
- [Configuring Identity Extension Properties Using Fusion Middleware Control](#)

- [Secure Conversation Configuration for the Domain Using Fusion Middleware Control](#)

14.5.1 OWSM Keystore Configuration Using Fusion Middleware Control

OWSM provides support for KSS, JKS, HSM, and PKCS11 keystores. After you create the keystores, you need to configure OWSM so that it can access and use the keystore.



Note:

There is one OWSM keystore per domain, and it is shared by all web services and clients running in the domain.

Note that there is one OWSM keystore per domain, and it is shared by all web services and clients running in the domain.

The following sections describe how to configure OWSM for each keystore type:

- [Configuring OWSM to Use the KSS Keystore](#)
- [Configuring OWSM to Use the JKS Keystore](#)
- [Configuring OWSM to Use HSM Keystores](#)
- [Configuring OWSM to Use the PKCS11 Keystore](#)

14.5.1.1 Configuring OWSM to Use the KSS Keystore

You can configure OWSM to use the KSS keystore using the Fusion Middleware Control.

To configure OWSM to use the KSS keystore, perform the following steps:

1. Create the keystore as described in "[Understanding OPSS Keystore Service for Message Protection](#)".
2. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
3. Select the **Message Security** tab.
4. In the Keystore section of the page, select `KSS` as the **Keystore Type**.

The KSS keystore settings are displayed, as shown in [Figure 14-4](#).

Figure 14-4 KSS Keystore Section of Message Security Page

The screenshot shows the 'Message Security' configuration page for 'base_domain'. Under the 'Key Store' section, the 'Keystore Type' is set to 'KSS'. The 'Path' field is empty. Below it, the 'Sign Alias' and 'Encrypt Alias' dropdown menus are also empty.

5. Enter the path to the keystore using the format `kss://stripeName/keystoreName`. For example: `kss://owsm/keystore`.

 **Note:**

This field is case sensitive.

6. Specify the sign and encrypt alias as follows:
 - For the **Sign Alias**, select the alias name from the menu. The value you select here must match the value in the keystore. For example, `orakey`.
 - For the **Encrypt Alias**, select the alias name from the menu. The value you select here must match the value in the keystore. For example, `orakey`.

 **Note:**

The alias names are only available in the menus if they exist in the keystore specified in the **Path** field. If the aliases do not exist in the keystore, the menu is blank.

7. Click **Apply** to accept the changes.
8. If you are configuring the keystore for the first time, a server restart is not required. If, however, you are changing the keystore or keystore configuration, you must restart the server.

 **Note:**

If you do not use the default keystore name for the KSS keystore (`kss://owsm/keystore`), you must grant permission to the `wsm-agent-core.jar` in OPSS. For more information about granting permissions, see "Setting the Java Security Policy Permission" in *Securing Applications with Oracle Platform Security Services*.

14.5.1.2 Configuring OWSM to Use the JKS Keystore

You can configure OWSM to use the JKS Keystore using Fusion Middleware Control.

When you configure OWSM to use the JKS keystore, entries are created in the credential store for the credential map `oracle.wsm.security`, and any keys that you define. (For details about configuring the credential store, see ["Adding Keys and User Credentials to Configure the Credential Store"](#).)

To configure OWSM to use the JKS keystore:

1. Create the keystore as described in ["Understanding Java Keystore for Message Protection"](#).
2. Navigate to the WSM Domain Configuration page as described in ["Navigating to the WSM Domain Configuration Page"](#).
3. Select the **Message Security** tab.
4. In the Keystore section of the page, select `JKS` as the **Keystore Type**.

The JKS keystore settings are displayed, as shown in [Figure 14-5](#).

Figure 14-5 JKS Section of Message Security Key Store Page

The screenshot shows the 'WSM Domain Configuration: base_domain' page with the 'Message Security' tab selected. Under the 'Key Store' section, the 'Keystore Type' is set to 'JKS (Java Key Store)'. Below this, there are three columns of input fields:

- Column 1:** Path, Key (dropdown), Password, Confirm.
- Column 2:** Key (dropdown), Sign Alias, Password, Confirm.
- Column 3:** Key (dropdown), Encrypt Alias, Password, Confirm.

5. In the left column, provide the path to the keystore, the keystore key, and the keystore password as follows:
 - In the **Path** field, enter the path and name for the keystore that you created as described in ["Generating Private Keys and Creating the Java Keystore"](#). For example, if you named the keystore `default-keystore.jks`, enter `./default-keystore.jks`. The location in this example is relative to the `domain_name/config/fmwconfig` directory. Alternatively, you can specify the absolute path to the keystore. If you used a different name or location for the keystore, enter those values instead.
 - Do one of the following:
 - If a keystore password key exists in the credential store, select it from the **Key** menu. If you select a key from the menu, the **Password** fields are disabled and the credentials associated with the selected key will be used.
 - If the keystore password key is not displayed in the **Key** menu, enter a keystore password key in the **Key** field. Note that you may need to clear the **Key** field before entering a new value. In the **Password** and **Confirm** fields, enter the password for the keystore. The key that you specify here will be stored in the credential store as the credential store key and the password that you enter will be associated with that key in the credential store. 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"](#), for example `password`. For more information, see ["How OWSM Locates Keystore And"](#)

Key Passwords for the JKS and PKCS11 Keystores" in *Understanding Oracle Web Services Manager*.

6. Configure the signature key as follows:
 - If a signature key exists in the credential store, select it from the **Key** menu. If you select a key from the menu, the **Sign Alias** and **Password** fields are disabled and the credentials associated with the selected key will be used.
 - If the signature key is not listed in the **Key** menu, enter a name for the signature key in the **Key** field. Note that you may need to clear the **Key** field before entering a new value. Specify an alias name in the **Sign Alias** field, and the password for the alias in the **Password** and **Confirm** fields. The values you specify here must match the values in the keystore. For example, `orakey` and `password`.
7. Configure the encryption key as follows:
 - If an encryption key exists in the credential store, select it from the **Key** menu. If you select a key from the menu, the **Encrypt Alias** and **Password** fields are disabled and the credentials associated with the selected key will be used.
 - If the encryption key is not listed in the **Key** menu, enter a name for the encryption key in the **Key** field. Note that you may need to clear the **Key** field before entering a new value. Specify an alias name in the **Encrypt Alias** field, and the password for the alias in the **Password** and **Confirm** fields. The values you specify here must match the values in the keystore. For example, `orakey` and `password`.

 **Note:**

The alias and password for the signature and encryption keys define the string alias and password used to store and retrieve the keys in the credential store. For more information, see "How OWSM Uses the Credential Store" in *Understanding Oracle Web Services Manager*.

8. Click **Apply** to submit the changes.
9. If you are configuring the keystore for the first time, a server restart is not required. If, however, you are changing the keystore or keystore configuration, you must restart the server.

14.5.1.3 Configuring OWSM to Use HSM Keystores

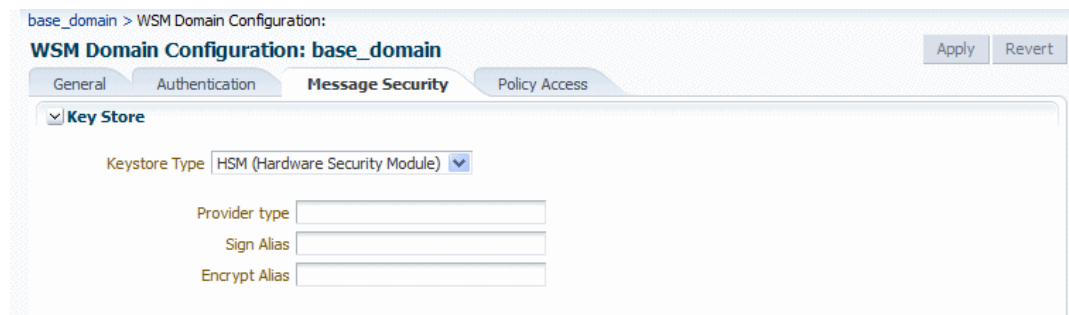
Hardware Security Modules (HSM), such as SafeNet Luna SA, can be integrated with OWSM to provide a secure way to store keys and off-load cryptographic processing. You can use Fusion Middleware Control to configure OWSM to use the HSM keystore.

To configure OWSM to use the HSM keystore:

1. Install and configure SafeNet Luna as described in "[Using Hardware Security Modules With OWSM](#)".
2. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
3. Select the **Message Security** tab.
4. In the Key Store section of the page, select `HSM` as the **Keystore Type**.

The HSM keystore settings are displayed, as shown in [Figure 14-6](#).

Figure 14-6 HSM Settings for OWSM Keystore Configuration



5. In the **Provider type** field, enter `luna`.
6. In the **Sign Alias** and **Encrypt 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.)

 **Note:**

For HSMs, only a signature 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 **Apply** to accept the changes.
8. If you are configuring the keystore for the first time, a server restart is not required. If, however, you are changing the keystore or keystore configuration, you must restart the server.

14.5.1.4 Configuring OWSM to Use the PKCS11 Keystore

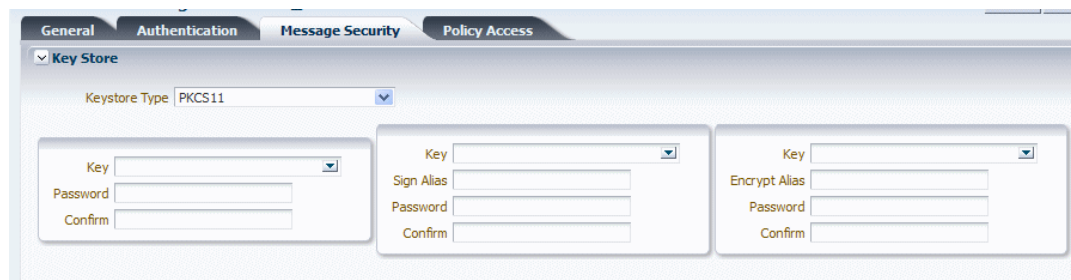
You can use Fusion Middleware Control to configure OWSM to use the PKCS11 keystore.

To configure OWSM to use the PKCS11 keystore:

1. Create the keystore as described in "[About Configuring OWSM for Oracle SPARC T5 and SPARC T4 Cryptographic Acceleration](#)".
2. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
3. Select the **Message Security** tab.
4. In the Keystore section of the page, select `PKCS11` as the **Keystore Type**.

The PKCS11 keystore settings are displayed, as shown in [Figure 14-7](#).

Figure 14-7 PKCS11 Section of Message Security Key Store Page



5. Do one of the following:
 - If a keystore password key exists in the credential store, select it from the **Key** menu. If you select a key from the menu, the **Password** fields are disabled and the credentials associated with the selected key will be used.
 - If the keystore password key is not listed in the **Key** menu, enter a keystore password key in the **Key** field. Note that you may need to clear the **Key** field before entering a new value. In the **Password** and **Confirm** fields, enter the password for the keystore. The key that you specify here will be stored in the credential store as the credential store key and the password that you enter will be associated with that key in the credential store. This password must match the password you used when you created the keystore, for example `password`. For more information, see "How OWSM Locates Keystore And Key Passwords for the JKS and PKCS11 Keystores" in *Understanding Oracle Web Services Manager*.
6. Configure the signature key as follows:
 - If a signature key exists in the credential store, select it from the **Key** menu. If you select a key from the menu, the **Sign Alias** and **Password** fields are disabled and the credentials associated with the selected key will be used.
 - If the signature key is not listed in the **Key** menu, enter a name for the signature key in the **Key** field. Note that you may need to clear the **Key** field before entering a new value. Specify an alias name in the **Sign Alias** field, and the password for the alias in the **Password** and **Confirm** fields. The values you specify here must match the values in the keystore. For example, `orakey` and `password`.
7. Configure the encryption key as follows:
 - If an encryption key exists in the credential store, select it from the **Key** menu. If you select a key from the menu, the **Encrypt Alias** and **Password** fields are disabled and the credentials associated with the selected key will be used.
 - If the encryption key is not listed in the **Key** menu, enter a name for the encryption key in the **Key** field. Note that you may need to clear the **Key** field before entering a new value. Specify an alias name in the **Encrypt Alias** field, and the password for the alias in the **Password** and **Confirm** fields. The values you specify here must match the values in the keystore. For example, `orakey` and `password`.

 **Note:**

The alias and password for the signature and encryption keys define the string alias and password used to store and retrieve the keys in the credential store. For more information, see "How OWSM Uses the Credential Store" in *Understanding Oracle Web Services Manager*.

8. Click **Apply** to submit the changes.
9. If you are configuring the keystore for the first time, a server restart is not required. If, however, you are changing the keystore or keystore configuration, you must restart the server.

14.5.2 Configuring Security Policy Enforcement Using Fusion Middleware Control

The Security Settings section of the **Message Security** tab allows you to tune the configuration of 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.

To configure security policy enforcement:

1. Navigate to the WSM Domain Configuration page. See [Navigating to the WSM Domain Configuration Page](#).
2. Select the **Message Security** tab.

Figure 14-8 Security settings in Message Security page

3. In the Security Settings section of the page, set the following properties as required for your environment:
 - **Allow XPath Transformations**—Specify whether OWSM will accept all types of XPath transformations. By default, OWSM only accepts the XPath transformation `ancestor-or-self::*[namespace-uri()='<namespace>' and local-name()='<name>']` inside the signature (in the incoming SOAP message).

Enable this property to allow and accept all types of XPath transformations. The default is disabled. Note that enabling this property may result in XPath based Denial of Service attacks or other similar XPath based security vulnerabilities.

- **Nonce Time To Live** – Specify the total time-to-live 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. The default value is 28800000 milliseconds (8 hours).

You should configure this property to:

- Decrease the time-to-live amount to ensure that there is less memory being consumed in the server because nonce is removed after the reduced time period.
- Increase the time-to-live amount so that nonce is cached longer in the server, thereby ensuring the uniqueness/freshness of the message for a longer time period.

To set this property, enter a new value in the field, or click the up/down arrows to increase or decrease the default value.

 **Note:**

For username token settings in a policy, 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 **Nonce Time to Live** to be equal to or greater than the value set for **Message Expiration Time**.

- **Clock Skew** – Specify the tolerance of time differences, in milliseconds, between client and server machines. For example, when timestamps are sent in a message to a service that follows a different time zone, this property allows for the specified time tolerance. The default value is 360000 milliseconds (6 minutes).

 **Note:**

This property must have a value greater than or equal to 1 second. If the value is less than 1 second, then the property does not work as expected.

To adjust the clock skew, enter a new value in the field or click the up/down arrows to increase or decrease the default value.

You should configure this property as follows:

- Increase the clock skew when the client and web service are running on different systems and their system clocks are not in sync, which could result in the service rejecting messages from the client, with an error indicating the timestamp validation failed. Increasing the clock skew accounts for the difference in clocks between the client and the web service. For example, if the difference between the web service clock and the client clock is 10 minutes, increase the Clock Skew on the system hosting the web service to 10 minutes (600000 milliseconds).
- Decrease the clock skew if you want to narrow the window in which the web service is willing to accept messages from clients to avoid replay attacks.
- **Message Expiration Time**— Specify the 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. It is also used to control the timestamp expiry window on the client side when the message is

created. The value specified here applies to all message protection and SAML assertion timestamp elements. The default value is 300000 milliseconds (5 minutes).

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 expiration time must be increased. The message expiration time is derived from the values of Message Expiration Time and the expiry time in the incoming message, and is the lesser of the two.

For example, if the server's Message Expiration Time is set to 5 minutes and the incoming message expiry time is 6 minutes, then the effective timestamp validation window is only 5 minutes and the incoming message is only be valid in that 5 minute window. In this case, you need to increase the Message Expiration Time at the service side. (Increasing the timestamp expiry on the incoming message will not fix the problem because the message expiration time is derived from both values and is the lesser of the two.)

On the other hand, if the server's Message Expiration 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.

 **Note:**

A higher value of the Message Expiration Time may lead to a security vulnerability.

- **Signature Cache Enable**—If enabled, OWSM will cache the primary signature of SOAP messages in the signature cache for a stipulated time. If a message comes in this time with the same signature, it will be rejected as a duplicate message. To get this functionality `include-timestamp` flag in the policy should be `true`. If it is not `true`, OWSM will not know that the coming message is duplicate, and the message will be accepted as genuine.

By default this property is disabled. You can enable it by selecting this check box. You can also enable it through WLST.

4. Click **Apply** to apply the property updates.

14.5.3 Configuring Identity Extension Properties Using Fusion Middleware Control

The properties in the Identity section of the **Message Security** 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 is published, you can also specify whether to ignore the hostname verification feature.

For more information about the service identity certificate extension and hostname verification, see "[Understanding Service Identity Certificate Extensions](#)".

**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 "[Overview of Configuring Keystores for Message Protection](#)".

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 "[Overview of Configuring Keystores for Message Protection](#)".

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. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
3. Select the **Message Security** tab.
4. In the Identity section of the page, set the following properties as required for your environment:
 - **Ignore Hostname Verification** – Specify 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`.
 - **Ignore Identity WSDL**— Specify whether to enable or disable the consumption of the X509 Certificate from a client-side WSDL for the 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`.
5. Click **Apply** to apply the property updates.

14.5.4 Secure Conversation Configuration for the Domain Using Fusion Middleware Control

You can use the Message Security tab on the WSM Domain Configuration page to configure secure conversation for the domain.

The following topics provide information to help you configure secure conversation:

- [About Secure Conversation](#)
- [Configuring Secure Conversation with Fusion Middleware Control](#)

14.5.4.1 About Secure Conversation

OWSM implements the Web Services Trust and Web Services Secure Conversation specifications, which together provide secure communication between web services and their clients. You can use WS-SecureConversation to increase the performance and security of your web services.

For more detailed information about secure conversation, see "Understanding Secure Conversation" in *Understanding Oracle Web Services Manager*.

WS-SecureConversation is configurable at the domain level, and at the policy level. If you use the predefined WS-SecureConversation policies provided with OWSM, WS-

SecureConversation is already enabled. However, many of the predefined assertion templates provide secure conversation functionality, and policies based on these assertion templates can be configured to use secure conversation. For details about configuring secure conversation at the policy level, see "[About Configuring Secure Conversation](#)".

14.5.4.2 Configuring Secure Conversation with Fusion Middleware Control

You can follow the procedure in this section to configure secure conversation at the domain level.

To configure secure conversation at the domain level:

1. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
2. Select the **Message Security** tab.
3. In the WS Secure Conversation section of the page, optionally configure the following properties as required for your environment:
 - **Secure Conversation Token Lifetime**—Specify the default number of milliseconds after which the secure conversation session should expire. The security context is shared by the client and web service for the lifetime of a communication session. This is the time after which the SCT is expired. The default is 1800000 milliseconds (30 minutes).
 - **Secure Conversation Token Lifetime for Reauthentication**—Specify the default number of milliseconds after which secure conversation of reauthenticate use cases should expire. The reauthenticate lifetime session should usually have a larger value because the session is shared across multiple users. When reauthentication is true and the session is shared among many users, you can set this to a larger value. The default is 28800000 milliseconds (8 hours).
 - **Encrypt RM protocol message body**—(Applies to web service client only.) Specify whether the WS-RM protocol messages should be encrypted. By default, this property is disabled. If enabled, the body of protocol request messages such as `createSequence()` and `terminateSequence()` are encrypted.

The response message body for protocol messages depends on the request message body: if the request message from the client is encrypted for protocol messages, the web service sends the response encrypted, and vice versa.
4. Click **Apply** to apply the property updates.

14.6 OWSM Policy Access Configuration Using Fusion Middleware Control

The **Policy Access** tab on the WSM Domain Configuration page provides the ability to configure the Policy Manager connection for auto-discovery, and whether SSL should be used for the connection. You can also manage the policy cache and configure high availability by tuning the retry logic.

The configuration details are described in the following sections:

- [Understanding Configuring the Policy Manager Connection Using Fusion Middleware Control](#)
- [About Refreshing Configuration Cache in OWSM Manually by using Fusion Middleware Control](#)

- [Configuring SSL for the Policy Manager Connection Using Fusion Middleware Control](#)
- [High Availability Configuration and Cache Management Using Fusion Middleware Control](#)

14.6.1 Understanding Configuring the Policy Manager Connection Using Fusion Middleware Control

The **Policy Manager** section of the Policy Access tab enables you to configure auto-discovery of the Policy Manager, specify an alternate Policy Manager URL and credential store key and credentials to access the Policy Manager, and modify the refresh configuration and inventory delay settings.

The following topics provide information to help you configure a Policy Manager connection:

- [About Auto-Discovery and Connecting to the Policy Manager](#)
- [Configuring a Connection to the Policy Manager Using Fusion Middleware Control](#)

14.6.1.1 About Auto-Discovery and Connecting to the Policy Manager

OWSM uses an auto-discovery feature to locate and connect to an OWSM Policy Manager. By default, the auto-discovery logic will always connect to the Policy Manager in the local domain using non-secure protocol. If the auto-discovery logic cannot connect to a Policy Manager using non-secure protocol because the non-secure port is disabled, it will attempt to connect to a Policy Manager using secure protocol. However, when using auto-discovery, you can specify whether to use SSL to secure the connection. If you do so, the auto-discovery logic will only connect to Policy Managers using secure protocol and will not try to connect to any Policy Managers deployed on non-SSL servers, even if the SSL-enabled server goes down.

Auto-discovery is supported only in WebLogic domains hosting an OWSM Policy Manager instance. For more information about auto-discovery, see "[Cross-Component Wiring for Auto-Discovery of Policy Manager](#)".

You may want to disable auto-discovery 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.

14.6.1.2 Configuring a Connection to the Policy Manager Using Fusion Middleware Control

To configure the Policy Manager connection:

1. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
2. Select the **Policy Access** tab and make the following changes in the Policy Manager section of the page.
3. Specify the credential store key to use to obtain a principal (and its credentials) authorized to access an OWSM Policy Manager instance. You should configure this property when:
 - You want to specify an explicit account to connect with the OWSM Policy Manager rather than the system account, `OracleSystemUser`, that is used by OWSM by default.

- The Authentication Provider and LDAP directory that is configured does not support system accounts used by Oracle WebLogic, but which OWSM 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 "[About Modifying the Default User](#)".

To specify the credential store CSF key, do one of the following:

- If the desired CSF key exists in the credential store, select it from the **PM CSF Key** menu. If you select a key from the menu, the **Credential** and **Password** fields are disabled and the credentials associated with the selected key will be used.
- If the CSF key is not listed in the **PM CSF Key** menu, enter a name for the CSF key in the **PM CSF Key** field. Note that you may need to clear the **PM CSF Key** field before entering a new value. In the **Credential** field, enter the username to be used to access the Policy Manager instance. In the **Password** and **Confirm** fields, enter the password for the username provided in the credential field.

If you do not specify a **PM CSF key**, then the `OracleSystemUser` principal (standard for WebLogic Server environments) is used.

4. Specify whether the auto-discovery feature of the Policy Manager should be used and, if so, whether SSL is required. If **Auto Discover** is enabled, the checkbox for specifying whether SSL should be used is active.

By default, auto-discovery is enabled and SSL is disabled. In this case, the auto-discovery logic always attempts to connect to the Policy Manager using non-secure protocol. If it is unable to do so because the non-secure port is disabled, it will attempt to connect to a Policy Manager using secure protocol.

To use auto-discovery with SSL, verify that **Auto Discover** is selected, then select **Use SSL Only**. If SSL is enabled, then the auto-discovery logic will only connect to Policy Managers using secure protocol and will not try to connect to any Policy Managers deployed on non-SSL servers, even if the SSL-enabled server goes down.

Auto-discovery is supported only in WebLogic domains hosting an OWSM Policy Manager instance.

To disable the auto-discovery feature, clear the **Auto Discover** checkbox. If disabled, the **Edit** icon is enabled and you can manually enter a URL to access a Policy Manager instance as described in the next step.

5. If auto-discovery is disabled, specify a URL that specifies the location of the Policy Manager. To do so:
 - a. Clear the **Auto Discover** checkbox, if you have not already done so.
 - b. Click **Edit** in the **PM URL** field.
 - c. In the Edit PM URLs window, click the **+** sign.
 - d. Enter the URL for the Policy Manager in the blank field.

The following protocols are valid:

`file://location_of_mds_home` — Repository is accessed directly as an MDS instance (supported only in Java SE).

`classpath://JAR_location` — JAR file containing the predefined policies and assertion templates is accessed using the classpath location.

 **Note:**

Because classpath mode is a read-only Policy Manager mode, only design time policy attachments will be in effect. No post-deployment policy attachments or global policy sets will be in effect in this mode.

If you do not provide a value in the URL field, OWSM uses auto-discovery in the same domain using non-secure protocol.

6. Adjust the **Refresh Rate** and the **Inventory Record Delay** if required:
 - The Refresh Rate specifies the number of milliseconds to wait between configuration refreshes. The default is 600,000 milliseconds (10 minutes).
 - The Inventory Record Delay specifies the number of milliseconds to wait before sending inventory data. The default is 60,000 milliseconds (1 minute).

Enter a new value in the field or click the up/down arrows to increase or decrease the default value.
7. Click **Apply** to apply the property updates.

14.6.2 About Refreshing Configuration Cache in OWSM Manually by using Fusion Middleware Control

OWSM has background threads to refresh configuration and documents cache automatically at regular time intervals. This interval is specified in the following properties: `refresh.repeat` and `cache.refresh.repeat`. You can disable this automatic refresh by setting the value of configuration property `auto.refresh` to `disabled`. Then, you can manually refresh the configuration and documents cache when required.

The following sections explain this further:

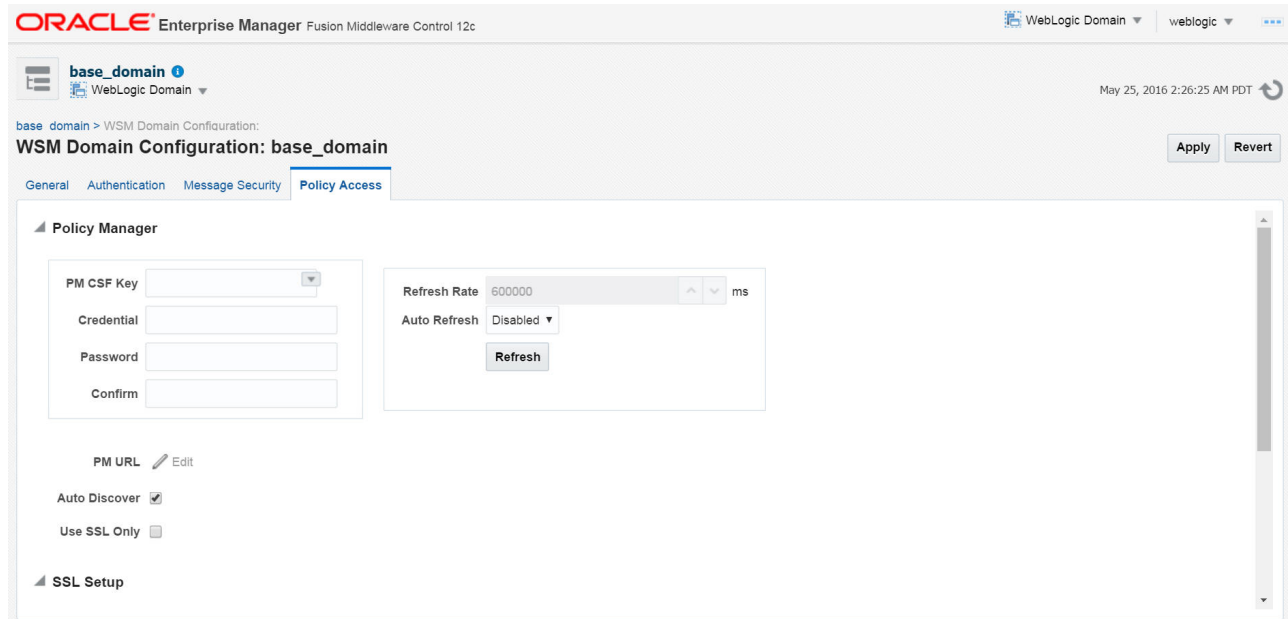
- [Disabling Automatic Refresh Option in OWSM by using Fusion Middleware Control](#)
- [Checking Status of Automatic Refresh in OWSM by using Fusion Middleware Control](#)
- [Refreshing the OWSM Cache Manually by using Fusion Middleware Control](#)

14.6.2.1 Disabling Automatic Refresh Option in OWSM by using Fusion Middleware Control

You can disable the automatic refresh option in OWSM by setting the value of configuration property **Auto Refresh** to **Disabled**. To disable the automatic refresh option:

1. Navigate to the WSM Domain Configuration page as described in [Navigating to the WSM Domain Configuration Page](#)
2. Select the **Policy Access** tab, and in the **Policy Manager** section of the page, set the value of the **Auto Refresh** to **Disabled**.

Figure 14-9 Disabling Auto Refresh Option to Manually Refresh Configuration Cache in OWSM



3. Click **Apply** to apply the changes.

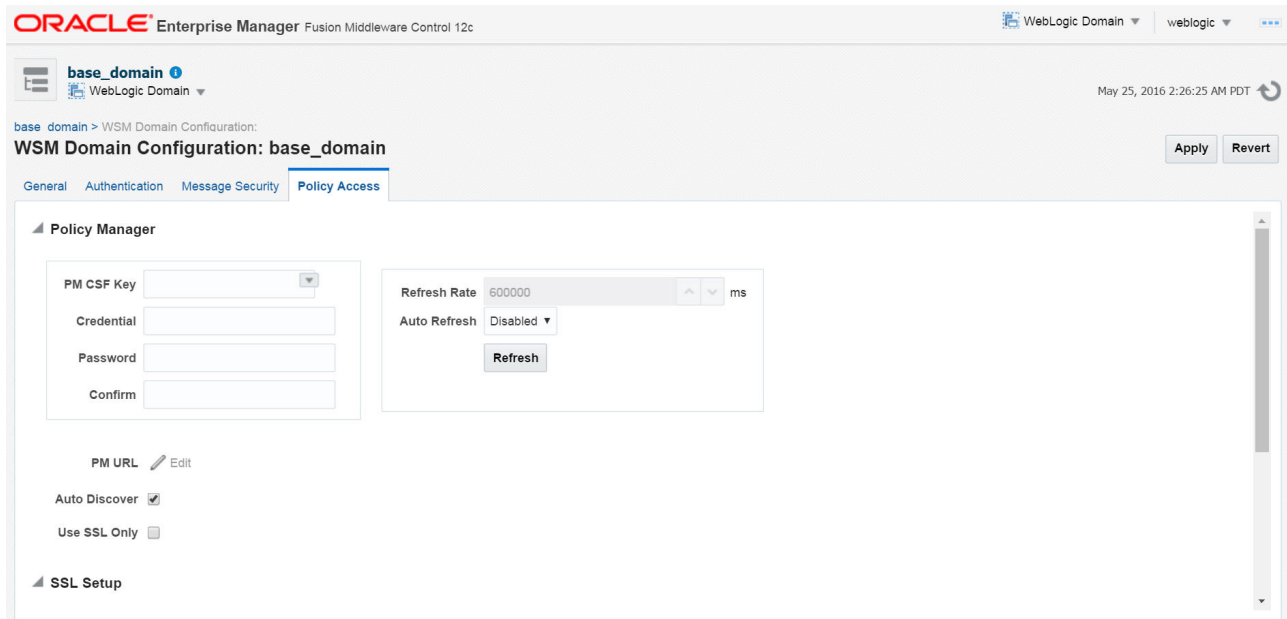
14.6.2.2 Checking Status of Automatic Refresh in OWSM by using Fusion Middleware Control

To check the status of the automatic refresh:

1. Navigate to the WSM Domain Configuration page as described in [Navigating to the WSM Domain Configuration Page](#)
2. Select the **Policy Access** tab.

The **Auto Refresh** option in the **Policy Manager** section displays the status of the automatic refresh in OWSM.

Figure 14-10 Policy Access Tab in WSM Domain Configuration Page Showing the Status of Auto Refresh

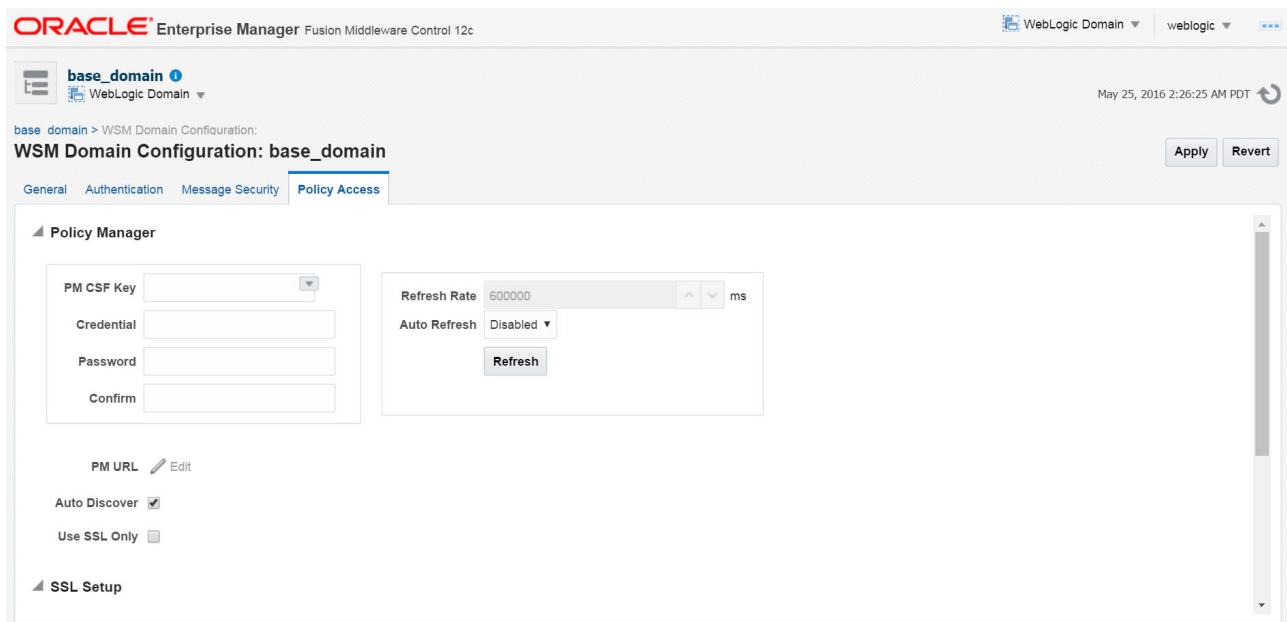


14.6.2.3 Refreshing the OWSM Cache Manually by using Fusion Middleware Control

To refresh the configuration and documents cache manually:

1. Navigate to the WSM Domain Configuration page as described in [Navigating to the WSM Domain Configuration Page](#)
2. Select the **Policy Access** tab.

Figure 14-11 Policy Access Tab in WSM Domain Configuration Page



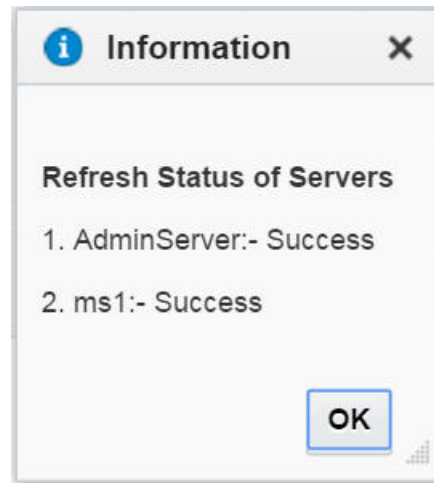
3. In the **Policy Manager** section, click **Refresh** to refresh the Configuration Cache in all the servers.

Note:

The manual refresh impacts refresh of OWSM configuration and documents only (like policy and policy sets).

The **Information** dialog box appears showing the refresh status of the servers.

Figure 14-12 Refresh Status of Servers



14.6.3 Configuring SSL for the Policy Manager Connection Using Fusion Middleware Control

The properties in the SSL Setup section of the Policy Access tab enable you to configure the type of SSL, if any, to be used for the communication between the OWSM runtime and the Policy Manager.

To configure SSL:

1. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
2. Select the **Policy Access** tab.
3. In the SSL Setup section of the page, specify the type of SSL to be used, if any. Choose one of the following options:
 - **Oneway**—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. If one-way SSL is enabled, you must provide the following credentials required to access the trust keystore:
 - **Truststore Path**—Enter a fully qualified path to the trust keystore.
 - **Key**—Select the CSF key to use to obtain the credential of the truststore. If the required key is not listed in the menu, you can enter it in this field. Note that you may have to clear the field before entering the new value.
 - **Password/Confirm**—Enter/reenter the password to access the truststore. These fields are disabled if you select an existing CSF key from the **Key** menu. In this case, the credentials associated with the selected key will be used

- **Two-way**—With two-way SSL, the server presents a certificate to the client and the client presents a certificate to the server. If two-way SSL is enabled, you must provide the following credentials for the truststore and the keystore:
 - **Truststore Path**—Enter a fully qualified path to the trust keystore.
 - **Key**—Select the CSF key to use to obtain the credential of the truststore. If the required key is not listed in the menu, you can enter it in this field. Note that you may have to clear the field before entering the new value.
 - **Password/Confirm**—Enter/reenter the password to access the truststore. These fields are disabled if you select an existing CSF key from the **Key** menu. In this case, the credentials associated with the selected key will be used.
 - **KeyStore**—Enter a fully qualified path to the Java keystore.
 - **Key**—Select the keystore key to use to obtain the credential of the keystore. If the required key is not listed in the menu, you can enter it in this field. Note that you may have to clear the field before entering the new value.
 - **Password/Confirm**—Enter/reenter the password to access the keystore. These fields are disabled if you select an existing CSF key from the **Key** menu and the credentials associated with the selected key will be used.
 - **SSL Alias**—Enter the keystore alias.
 - **None**—Do not use SSL.
4. Click **Apply** to apply the property updates.

14.6.4 High Availability Configuration and Cache Management Using Fusion Middleware Control

The properties in the Cache Management section of the **Policy Access** tab enable you to manage the policy cache and configure high availability by tuning the retry logic.

The following topics provide information to help you configure high availability and manage the policy cache:

- [About High Availability and Cache Management](#)
- [Configuring High Availability and Managing the Cache Using Fusion Middleware Control](#)

14.6.4.1 About High Availability and Cache Management

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. You can set this property as described in "[Configuring the Policy Manager Connection Using WLST](#)".

If the initial connection is made, but the OWSM 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 many 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 **Initial Cache Refresh** and **Cache Refresh Time** 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 Documents 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 Documents Retry Delay** property will still affect how often it attempts to communicate with the Policy Manager until it is fixed.

14.6.4.2 Configuring High Availability and Managing the Cache Using Fusion Middleware Control

To configure high availability and manage the policy cache:

1. Navigate to the WSM Domain Configuration page as described in "[Navigating to the WSM Domain Configuration Page](#)".
2. Select the **Policy Access** tab.
3. In the Cache Management section of the page, configure the following properties as required for your environment:
 - **Failure Retry Delay**—Specify the number of milliseconds to wait between retry attempts. The default is 5000 milliseconds.
 - **User Record Delay**—Specify the number of milliseconds to wait before sending usage data. The default is 30000 milliseconds.
 - **Failure Retry Count**—Specify the number of times to retry after a communication failure. The default is 2 retry attempts.
 - **Missing Documents Retry Delay**—Specify the number of milliseconds to wait before trying to retrieve a missing document. The default is 15000 milliseconds.
 - **Initial Cache Refresh**—Specify the number of milliseconds to wait before initial cache refresh. The default is 600000 milliseconds (10 minutes).
 - **Cache Refresh Time**—Specify the number of milliseconds to wait between cache refreshes. The default is 600000 milliseconds (10 minutes).

To set these properties, enter a new value in the field, or click the up/down arrows to increase or decrease the default value.

4. Click **Apply** to apply the property updates.

14.7 About Managing OWSM Domain Configuration Properties Using WLST

You can use WLST to view and set domain-level configuration properties for OWSM, including authentication, message security, and policy access. You can set most of these configuration properties using the `setWSMConfiguration` command.

The following sections provide details:

- [Viewing OWSM Domain Configuration Using WLST](#)
- [Setting OWSM Domain Configuration Properties Using the `setWSMConfiguration` Command](#)

Additional commands are provided to configure trusted issuers and the message protection keystore. Procedures for using those commands are provided in the following sections:

- [Configuring SAML and JWT Trusted Issuers, DN Lists, and Token Attribute Rules Using WLST](#)

- [Deleting a Token Issuer Trust Document Using WLST](#)
- [Configuring the OWSM Keystore Using WLST](#)

For reference information about these WLST commands, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

14.7.1 Viewing OWSM Domain Configuration Using WLST

You can display all of the configuration properties, with their values and groups, as specified in the current configuration document in the repository. If a property is not defined in the configuration document, then the default value defined for the product is displayed.

To view the OWSM domain configuration:

1. Connect to the running instance of the server in the domain for which you want to view the configuration as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Use the `displayWSMConfiguration()` command to display the OWSM domain configuration.

```
displayWSMConfiguration([context=None])
```

For example, to display the configuration for the domain specified by the context `wls/base_domain`, enter the following command:

```
wls:/new_domain/serverConfig> displayWSMConfiguration('/wls/base_domain')
```

```
NAME: "allow.all.xpathes" CATEGORY: "Agent" SOURCE: "default"Value: false
NAME: "use.unified.fault.code" CATEGORY: "Agent" SOURCE: "default"Value: true
NAME: "client.clock.skew" CATEGORY: "Agent" SOURCE: "default"Value: 0
NAME: "compliance.check" CATEGORY: "Agent" SOURCE: "default"Value: true
NAME: "clock.skew" CATEGORY: "Agent" SOURCE: "default"Value: 360000
NAME: "expire.time" CATEGORY: "Agent" SOURCE: "default"Value: 300000
.
.
.
```

Note that the output displayed above shows the source of the configuration property settings as `default`, indicating the setting is from the default configuration document for the product. If you have modified a setting, it is saved in a configuration document for your domain, and will be reflected in the `SOURCE` field. For example:

```
NAME: "remove.anonymous.role" CATEGORY: "SubjectProperties" SOURCE: "/WLS/base_domain"
Value: true
```

14.7.2 Setting OWSM Domain Configuration Properties Using the `setWSMConfiguration` Command

You can set the domain-level configuration properties using the `setWSMConfiguration` command.

You do not need to use the `setWSMConfiguration` command in the context of a session.

To modify the configuration properties for the OWSM domain using the `setWSMConfiguration` command:

1. Connect to the running instance of the server in the domain to be configured as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Optionally, use the `displayWSMConfiguration()` command to view the current configuration for the domain as described in "[Viewing OWSM Domain Configuration Using WLST](#)".
3. Use the `setWSMConfiguration()` command to set the desired configuration properties.

```
setWSMConfiguration (context,category,name,[group=None],[values=None])
```

In this command:

- `context`—Specifies the context of the configuration document to be modified. If a context is not provided or is set to `None`, then the configuration document associated with the currently connected domain is used.
- `category`—The category of the property. The category is verified against the default set of properties to ensure it is acceptable for the context. This field is required.
- `name`—The name of the property. The name is verified against the default set of properties to ensure it is acceptable for the context. This field is required.
- `group`—The group containing the set of values to add in the configuration document. If the group exists, and this value is set to `None`, the group is removed. This field is optional.
- `values`—The array of values to set for a property or group in the configuration document. This field is optional.

For example, to modify the default SAML2 login module properties, you can specify the following commands:

```
wls:/base_domain/serverConfig>
setWSMConfiguration(None,'SAML2LoginModule','add.assertion.to.subject',None,
['false'])
```

```
A new property "add.assertion.to.subject" within category "SAML2LoginModule" has
been added.
The values "[false]" have been added to property "add.assertion.to.subject" within
category "SAML2LoginModule".
Configuration properties associated with the context "/wls/base_domain" has been
created.
```

```
wls:/base_domain/serverConfig>
setWSMConfiguration(None,'SAML2LoginModule','allow.virtual.user',None,['false'])
```

```
A new property "allow.virtual.user" within category "SAML2LoginModule" has been
added.
The values "[false]" have been added to property "allow.virtual.user" within
category "SAML2LoginModule".
Configuration properties associated with the context "/WLS/base_domain" has been
updated.
```

Refer to the subsequent sections for the list of configuration properties for each type of configuration.

14.8 Configuring Domain-Level Authentication Using WLST

You can use WLST to set domain level authentication properties for OWSM, including the ability to configure SAML trust, JWT trust, the lifetime to be used for the issued token, and subject properties for JAAS subjects that are created after OWSM authentication. You can also configure SAML, SAML2, Kerberos, X509, and custom login modules.

The configuration details are described in the following sections:

- [Configuring SAML and JWT Trusted Issuers, DN Lists, and Token Attribute Rules Using WLST](#)
- [Deleting a Token Issuer Trust Document Using WLST](#)
- [Configuring the Lifetime for the Issued Token Using WLST](#)
- [Configuring the SAML and SAML2 Login Modules Using WLST](#)
- [Configuring the Kerberos Login Module Using WLST](#)
- [Configuring Subject Properties Using WLST](#)
- [Configuring the X509 Login Module Using WLST](#)
- [Configuring Custom Login Modules Using WLST](#)

14.8.1 Configuring SAML and JWT Trusted Issuers, DN Lists, and Token Attribute Rules Using WLST

SAML and JWT trusted issuers and DN lists are stored in trust configuration documents in the OWSM repository. To configure trusted issuers and DN lists, you must create a new document or edit an existing document in the repository.

When using WLST to create, modify, and delete token issuer trust documents, you must execute the commands in the context of a session. Each session applies to a single trust document only.

For more information about trusted issuers, DN lists, and token attribute rules, see the following topics:

- ["SAML Trusted Issuers and DN Lists Using Fusion Middleware Control"](#).
- ["Configuring Token Attribute Rules for Trusted Issuers Using Fusion Middleware Control"](#)
- ["Configuring JWT Trusted Issuers and DN Lists Using Fusion Middleware Control"](#)

To configure SAML and JWT trusted issuers, DN lists, and token attribute rules using WLST:

1. Connect to the running instance of the server in the domain for which you want to view the configuration as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Start a repository session using the `beginWSMSession` command.

The `beginWSMSession` 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:/base_domain/serverConfig> beginWSMSession()
```

Session started for modification.

3. List the token issuer trust documents in the repository using the `listWSMTokenIssuerTrustDocuments` command.

```
listWSMTokenIssuerTrustDocuments(name='*', detail='false')
```

When used without any arguments, all the token issuer trust documents in the repository are listed. If you set the `detail` argument to `true`, the display name and the status of the document are also displayed.

For example:

```
wls:/base_domain/serverConfig> listWSMTokenIssuerTrustDocuments(detail='true')
```

List of Token Issuer Trust Documents in the Repository:

```
Name           : oracle-default
Display Name   : il8n:oracle.wsm.resources.resdesc.ResourceDescriptionBundle_property-
TokenIssuerTrust_displayName
Status        : DOCUMENT_STATUS_COMMITTED
```

4. Do one of the following:

- If a token issuer trust domain document does not exist for your domain, create one using the `createWSMTokenIssuerTrustDocument` command. The `name` argument is required. Typically, the name of the document should use the following format: `tokenissuertrust<ServerType><domainName>`, for example, `tokenissuertrustWLSbase_domain`.

```
createWSMTokenIssuerTrustDocument(name, displayName)
```

For example:

```
wls:/base_domain/serverConfig>
createWSMTokenIssuerTrustDocument('tokenissuertrustWLSbase_domain')
```

New Token Issuer Trust document named "tokenissuertrustWLSbase_domain" created. Run the `setWSMConfiguration` command where `category = "TokenIssuerTrust"`, `property name = "name"` and `value = "tokenissuertrustWLSbase_domain"`, for the new document to be used in the domain configuration.

```
wls:/base_domain/serverConfig> setWSMConfiguration
(None, 'TokenIssuerTrust', 'name', None, ['tokenissuertrustWLSbase_domain'])
```

The values "[tokenissuertrustWLSbase_domain]" have been added to property "name" within category "TokenIssuerTrust". Configuration properties associated with the context "/WLS/base_domain" has been updated.

- If the token issuer trust document already exists for your domain, select the document for modification using the `selectWSMTokenIssuerTrustDocument` command. The `name` argument is required.

```
selectWSMTokenIssuerTrustDocument(name)
```

For example:

```
wls:/base_domain/serverConfig>
selectWSMTokenIssuerTrustDocument('tokenissuertrustWLSbase_domain')
```

Token Issuer Trust document named "tokenissuertrustWLSbase_domain" selected in the session.

- Optionally, specify a display name for the document using the `setWSMTokenIssuerTrustDisplayName` command. The display name is optional but can be useful for describing the documents. Note that if you specified a display name for the document when you created it, you can use this command to change the display name if desired.

```
setWSMTokenIssuerTrustDisplayName(displayName)
```

For example:

```
wls:/base_domain/serverConfig> setWSMTokenIssuerTrustDisplayName('base_domain Trust Document')
```

Display Name of the document changed from null to base_domain Trust Document.

- Add the trusted issuers and define trusted keys or a trusted DN list using the `setWSMTokenIssuerTrust` command.

```
setWSMTokenIssuerTrust(type, issuer, [trustedKeys=None])
```

In this command:

- `type` indicates the types of the tokens issued by the issuer and how the issuer signing certificates are identified with `trustedKeys`. 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.alias.sv	SAML SV	X509 certificate	alias
dns.alias.hok	SAML HOK or Bearer	X509 certificate	alias
dns.jwt	JWT	X509 certificate	DN

- `issuer` is the name of the trusted issuer, such as `www.oracle.com`.
- `trustedKeys` is an optional argument used to specify the trusted key identifiers or DN list or aliases 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 or aliases for the `trustedKeys` argument, the previous list is replaced with the new list. If you enter an empty set (`[]`) for the `trustedDNs` 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 `trustedKeys` argument, the issuer is created with the associated DN list. If you do not set the `trustedKeys` argument, a new issuer is created with an empty DN list.

In the following example, `www.example.com` is set as a trusted issuer. A DN list is not specified:

```
wls:/base_domain/serverConfig> setWSMTokenIssuerTrust("dns.sv", "www.example.com", [])
```

New issuer - "www.example.com" added to the document.

The issuer and trusted DN values have been updated successfully.

In the following example, 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:

```
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'])
```

```
New issuer - "www.oracle.com" added to the document.
Issuer set with the given trusted keys.
The issuer and trusted DN values have been updated successfully.
```

7. 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.

For example, to view the DN lists for the www.oracle.com trusted issuer:

```
wls:/base_domain/serverConfig> displayWSMTokenIssuerTrust('dns.sv', 'www.oracle.com')
```

```
List of trusted key(s) for this issuer:
```

```

Key Identifier : CN=weblogic, OU=Orakey Test Encryption
Purposes Only, O=Oracle, C=US
Key Type       : x509certificate
Value Type     : dn

Key Identifier : CN=orcladmin, OU=Doc, O=Oracle, C=US
Key Type       : x509certificate
Value Type     : dn
```

To view all of the trusted issuers for the type dns.sv:

```
wls:/base_domain/serverConfig> displayWSMTokenIssuerTrust('dns.sv')
```

```
List of trusted issuers for this type:
```

```
www.example.com
www.oracle.com
```

```
List of trusted key(s) for this issuer:
```

```

Key Identifier : CN=weblogic, OU=Orakey Test Encryption
Purposes Only, O=Oracle, C=US
Key Type       : x509certificate
Value Type     : dn

Key Identifier : CN=orcladmin, OU=Doc, O=Oracle, C=US
Key Type       : x509certificate
Value Type     : dn
```

8. Optionally, add additional security constraints by using the setWSMTokenIssuerTrustAttributeFilter command to define token attribute rules for a trusted DN. The attribute rule can be applied to a subject name ID.

```
setWSMTokenIssuerTrustAttributeFilter(dn, attr-name, [filters])
```

In this command:

- dn represents the DN of the token signing certificate.

- `attr-name` represents the name of the attribute to assert, using the format `name-id` for the subject name ID.
- `filters` represents the list of filters for the attribute with the format `['value1', 'value2', 'value3' ...]`. Each value can be an exact name or a name pattern with a wildcard `"*"` character. When `name-id` is specified for the `attr-name` argument, then the value of the subject name ID in the incoming SAML assertion must match one of the specified values to go through. If no values are specified, then any value for the subject name ID will go through.

This command behaves as follows:

- If the attribute specified by the `attr-name` argument already exists with a list of filter values and you provide a new list of values for the `filters` argument, the previous list is replaced with the new list. If you enter an empty set `([])` for the `filters` argument, then the existing list of filter values is deleted.
- If the attribute specified by the `attr-name` argument does not exist and you specify a list of values for the `filters` argument, the attribute is created and added to the document with the specified filter values. If you do not provide a value for the `filters` argument, an error is thrown.

In this example, the name IDs `yourTrustedUser` and `jdoh` are set as trusted users for the `weblogic` trusted DN:

```
wls:/base_domain/serverConfig> setWSMTokenIssuerTrustAttributeFilter('CN=weblogic,
OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US',
'name-id', ['yourTrustedUser', 'jdoh'])
```

New TokenAttributeRule added for DN: CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US.

9. Optionally, delete a token attribute rule using the `deleteWSMTokenIssuerTrustAttributeRule` command.

```
deleteWSMTokenIssuerTrustAttributeRule(dn)
```

For example, to delete the token attribute rule associated with the `weblogic` trusted DN:

```
wls:/base_domain/serverConfig> deleteWSMTokenIssuerTrustAttributeRule('CN=weblogic,
OU=Orakey Test Encryption Purposes Only,
O=Oracle, C=US')
```

Token Attribute Rule(s) for DN: CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US, deleted successfully.

Note:

This command deletes the attribute rule. To delete only the list of filter values for an attribute in an attribute rule, use the `setWSMTokenIssuerTrustAttributeFilter` command and enter an empty set `([])` for the `filters` argument.

10. Write the current contents of this session to the OWSM Repository using the `commitWSMSession` command.

For example:

```
wls:/base_domain/serverConfig> commitWSMSession()
The tokenissuertrust tokenissuertrustWLSbase_domain is valid.
```

```
Creating tokenissuertrust tokenissuertrustWLSbase_domain in repository.  
Session committed successfully.
```

Alternatively, you can choose to cancel any changes by using the `abortWSMSession` command, which discards any changes that were made to the repository during the session.

For more information about these commands, see "Token Issuer Trust Configuration Commands" in *WLST Command Reference for Infrastructure Components*.

14.8.2 Deleting a Token Issuer Trust Document Using WLST

Follow the procedure in this section to delete an token issuer trust document from the repository.

To delete an token issuer trust document from the repository:

1. Connect to the running instance of the server in the domain for which you want to view the configuration as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Start a repository session using the `beginWSMSession` command.

For example:

```
wls:/base_domain/serverConfig> beginWSMSession()
```

```
Session started for modification.
```

3. List the token issuer trust documents in the repository using the `listWSMTokenIssuerTrustDocuments` command.

```
listWSMTokenIssuerTrustDocuments(name='*', detail='false')
```

When used without any arguments, all the token issuer trust documents in the repository are listed. If you set the `detail` argument to `true`, the display name and the status of the document are also displayed.

For example:

```
wls:/base_domain/serverConfig> listWSMTokenIssuerTrustDocuments(detail='true')
```

```
List of Token Issuer Trust Documents in the Repository:
```

```
Name           : oracle-default  
Display Name   : i18n:oracle.wsm.resources.resdesc.ResourceDescriptionBundle_property-  
TokenIssuerTrust_displayName  
Status        : DOCUMENT_STATUS_COMMITTED
```

```
Name           : tokenissuertrustWLSbase_domain
```

```
Display Name   : base_domain Trust Document
```

```
Status        : DOCUMENT_STATUS_COMMITTED
```

```
List of Token Issuer Trust Documents in the Repository:
```

4. Delete the desired token issuer trust document using the `deleteWSMTokenIssuerTrustDocument` command. The `name` argument is required.

```
deleteWSMTokenIssuerTrustDocument(name)
```

For example:

```
wls:/base_domain/serverConfig>
```

```
deleteWSMTokenIssuerTrustDocument('tokenissuertrustWLSbase_domain')
```

```
Token Issuer Trust document named "tokenissuertrustWLSbase_domain" deleted from the  
repository.
```

5. Write the contents of the current session to the repository using the `commitWSMSession` command.

```
wls:/base_domain/serverConfig> commitWSMSession()
```

```
Deleting tokenissuertrust tokenissuertrustWLSbase_domain from repository.
```

```
Session committed successfully.
```

Alternatively, you can choose to cancel any changes by using the `abortWSMSession` command, which discards any changes that were made to the repository during the session.

14.8.3 Configuring the Lifetime for the Issued Token Using WLST

You can specify the lifetime to be used for the issued token when the request message is sent to the Security Token Service (STS).

If the STS sends the token with a different expiry, the runtime looks at the actual expiry of the token and only caches the token for that time. For more information, see "[Understanding Token Lifetime and Token Caching](#)".

To configure the lifetime of the issued token using WLST, use the `setWSMConfiguration` command as described in "[Setting OWSM Domain Configuration Properties Using the setWSMConfiguration Command](#)".

Use the following settings for this configuration property:

- Category: IssuedToken
- Name: lifetime
- Default: 28800000ms (8 hours)

For example, to change the default value to 25200000 ms (7 hours), use the following command:

```
wls:/base_domain/serverConfig> setWSMConfiguration (None,'IssuedToken','lifetime',None, [ '25200000' ])
```

```
A new property "lifetime" within category "IssuedToken" has been added.
```

```
The values "[25200000]" have been added to property "lifetime" within category "IssuedToken".
```

```
Configuration properties associated with the context "/WLS/base_domain" has been updated.
```

14.8.4 Configuring Subject Properties Using WLST

This topic describes the OWSM domain-level configuration properties that you can set for the subject properties. To configure subject properties for the domain, use the `setWSMConfiguration` command.

This command is described in "[Setting OWSM Domain Configuration Properties Using the setWSMConfiguration Command](#)".

For more information about subject properties, see "[Configuring Subject Properties Using Fusion Middleware Control](#)".

[Table 14-1](#) lists the OWSM domain-level configuration properties that you can set for the subject properties.

Table 14-1 Subject Domain Configuration Properties

Category	Property Name	Default	Description
SubjectProperties	add.application.roles	true	Flag that specifies if the application roles will be added to the subject created in OWSM. If <code>true</code> , the application roles are added to the subject. If set to <code>false</code> , then the application roles are not added to the subject.
SubjectProperties	add.authenticated.role	true	Flag that specifies if the authenticated role will be added to the Subject created in OWSM. If <code>true</code> , the authenticated role is added to the subject. If set to <code>false</code> , the authenticated role is not added to the subject.
SubjectProperties	remove.anonymous.role	false	Flag that specifies if the anonymous role will be removed from the subject created in OWSM. If <code>false</code> , then the anonymous role is added to the subject. If set to <code>true</code> , the anonymous role is removed from the subject.

14.8.5 Configuring the SAML and SAML2 Login Modules Using WLST

You can use the `setWSMConfiguration` command to configure the SAML and SAML2 login modules by using WLST.

The command is described in "[Setting OWSM Domain Configuration Properties Using the `setWSMConfiguration` Command](#)".

For more information about these login modules, see "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)".

[Table 14-2](#) lists the OWSM domain-level configuration properties that you can set for the SAML and SAML2 login modules.

Table 14-2 SAML and SAML2 Login Module Domain Configuration Properties

Category	Property Name	Default Value	Description
SAML2LoginModule SAMLLoginModule	add.assertion.to.subject	true	Flag that specifies if the SAML assertion should be added to the authenticated subject as a private credential. If you set this property to <code>false</code> , then the assertion is not added to the authenticated subject.

Table 14-2 (Cont.) SAML and SAML2 Login Module Domain Configuration Properties

Category	Property Name	Default Value	Description
SAML2LoginModule SAMLLoginModule	allow.virtual.user	false	Flag that specifies that the SAML subject is allowed to be treated as a virtual user. If set to <code>true</code> , 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. If this property is set to <code>false</code> , 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.
SAML2LoginModule SAMLLoginModule	dn.mapping.attribute	CN	Part of the certificate DN to use for asserting the user against the identity store if the Name Identifier Format is set to <code>X509SubjectName</code> in the SAML client policy.
SAML2LoginModule SAMLLoginModule	custom	N/A	Use this property to set custom properties to the OPSS login module.
SAMLLoginModule	dns.hok	www.oracle.com	Defines the list of trusted issuers for the <code>dns.hok</code> token. The DN of the endorser of the SAML token will be checked in the SAML issuer list. Use this property for SAML HOK and SAML bearer.
SAMLLoginModule	dns.sv	www.oracle.com	Defines the list of trusted issuers for the <code>dns.sv</code> token. The DN of the endorser of the SAML token will be checked in the SAML issuer list. Use this property for SAML sender vouches.

14.8.6 Configuring the Kerberos Login Module Using WLST

You can use the `setWSMConfiguration` command to configure the Kerberos login module using WLST.

The command is described in "[Setting OWSM Domain Configuration Properties Using the `setWSMConfiguration` Command](#)".

[Table 14-3](#) lists the OWSM domain-level configuration properties that you can set for the Kerberos login module.

Table 14-3 Kerberos Login Module Domain Configuration Properties

Category	Property Name	Default	Description
KerberosLoginModule	class name	N/A	The login module to be used by OWSM. If no value is specified, at runtime one of the following default values will be used for the Kerberos login module Class Name, depending on the platform on which you are running: <ul style="list-style-type: none"> <code>com.sun.security.auth.module.Krb5LoginModule</code> (Oracle) <code>com.ibm.security.auth.module.Krb5LoginModule</code> (IBM)
KerberosLoginModule	do.not.prompt	true	Specifies whether you will be prompted for the password if credentials cannot be obtained from the cache or keytab. When set to <code>true</code> , authentication will fail if credentials can not be obtained from the cache or keytab.
KerberosLoginModule	key.tab	./krb5.keytab	The file name of the keytab to get the secret key for the principal
KerberosLoginModule	principal	HOST/ localhost@EXAM PLE.COM	The name of the principal to be used. The principal represents a specific entity to which a set of credentials are assigned. It can be a simple username, such as <code>testuser</code> , or a service name such as <code>HOST/localhost</code> . You can use the <code>principal</code> option to set the principal when there are credentials for multiple principals in the keytab or when you want a specific ticket cache only.
KerberosLoginModule	store.key	true	Specifies whether the principal's key is stored in the Subject's private credentials.
KerberosLoginModule	use.key.tab	true	Specifies whether the module will get the principal's key from the keytab. If the 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 <code><user.home><file.separator>krb5.keytab</code> .
KerberosLoginModule	custom	N/A	Use to provide custom properties to the OPSS login module. You can set different groups in this property and these groups will have a single value.

14.8.7 Configuring the X509 Login Module Using WLST

You can use the `setWSMConfiguration` command to configure the X509 login module.

The command is described in "[Setting OWSM Domain Configuration Properties Using the `setWSMConfiguration` Command](#)".

[Table 14-4](#) lists the domain-level configuration properties that you can set for the X509 login module.

Table 14-4 X509 Login Module Domain-Level Configuration Properties

Category	Property Name	Default	Description
X509LoginModule	dn.mapping.attribute	CN	Specifies which part of the certificate DN to use for asserting the user against the identity store. The value that you specify is passed to OPSS with each invocation of the login module.
x509LoginModule	custom	N/A	Use to provide custom properties to the OPSS login module. You can set different groups in this property and these groups will have a single value.

14.8.8 Configuring Custom Login Modules Using WLST

You can configure other login modules, and create custom login modules in the OWSM configuration system using the `setWSMConfiguration` command.

The command is described in "[Setting OWSM Domain Configuration Properties Using the setWSMConfiguration Command](#)".

For details about creating a login module, see the *Java Authentication and Authorization Service (JAAS) LoginModule Developer's Guide* at <http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASLMDevGuide.html>.

[Table 14-5](#) lists the configuration properties for other login modules that are provided by default.

Table 14-5 Other Login Module Domain Configuration Properties

Category	Property Name	Default	Description
DigestLoginModule	custom	N/A	Use to provide custom properties to the OPSS login module. You can set different groups in this property and these groups will have a single value.
UsernameAssertionLoginModule	custom	N/A	Use to provide custom properties to the OPSS login module. You can set different groups in this property and these groups will have a single value.
UsernameAuthLoginModule	custom	N/A	Use to provide custom properties to the OPSS login module. You can set different groups in this property and these groups will have a single value.
WSSDigestLoginModule	custom	N/A	Use to provide custom properties to the OPSS login module. You can set different groups in this property and these groups will have a single value.

14.9 About Configuring Domain-Level Message Security Using WLST

You can use WLST to set domain-level message security properties for OWSM, including the ability to configure the OWSM keystore, tune security policy enforcement, specify if the public certificate should be published in the WSDL and the hostname verified, and secure conversation.

The configuration details are provided in the following sections:

- [Configuring the OWSM Keystore Using WLST](#)
- [Configuring Security Policy Enforcement Using WLST](#)
- [Configuring Identity Extension Properties Using WLST](#)
- [Configuring Secure Conversation for the Domain Using WLST](#)

14.9.1 Configuring the OWSM Keystore Using WLST

OWSM provides support for KSS, JKS, HSM, and PKCS11 keystores. After you create the keystores, you need to configure OWSM so that it can access and use the keystore. You can configure the OWSM keystore using the `configureWSMKeystore` command.

Note that there is one OWSM keystore per domain, and it is shared by all web services and clients running in the domain.

To configure the OWSM keystore:

1. Create the desired keystore as described in the following sections:
 - KSS: "[Understanding OPSS Keystore Service for Message Protection](#)".
 - JKS: "[Understanding Java Keystore for Message Protection](#)".
 - HSM: "[Using Hardware Security Modules With OWSM](#)".
 - PKCS11: "[About Configuring OWSM for Oracle SPARC T5 and SPARC T4 Cryptographic Acceleration](#)".
2. Connect to the running instance of the server in the domain to be configured as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
3. Optionally, use the `displayWSMConfiguration()` command to view the current configuration for the domain as described in "[Viewing OWSM Domain Configuration Using WLST](#)".

The keystore configuration properties are in the `KeystoreConfig` category.

4. Use the `configureWSMKeystore` command to configure the OWSM keystore properties.

```
configureWSMKeystore(context, keystoreType, location, keystorePassword, signAlias,
signAliasPassword, cryptAlias, cryptAliasPassword)
```

In this command:

- `context` represents the configuration document to be modified. If a configuration document associated with the context does not exist, it is created automatically.
- `keystoreType` represents the keystore type category of the property. Valid keystore types are JKS, KSS, PKCS11, and LUNA. The default is KSS.
- `location` represents the location of the keystore. For JKS, it is the absolute location of the keystore or the location relative to the `fmwconfig` directory. For KSS, the format of the location should be `kss://stripeName/keystoreName`. The default is `kss://owsm/keystore`. This argument is not required for LUNA and PKCS11 keystores.
- `keystorePassword` represents the keystore password for the configured keystore. It is required for JKS and PKCS11 keystores.
- `signAlias` represents the alias of the signature key. The value that you specify here must match the value in the keystore. For example, `orakey`.

- `signAliasPassword` represents the password for the signature key alias. It is required for JKS and PKCS11 keystores.
- `cryptAlias` represents the alias of the encryption key. The value that you specify here must match the value in the keystore. For example, `orakey`
- `cryptAliasPassword` represents the password for the encryption key alias. It is required for JKS and PKCS11 keystores.

 **Note:**

This command sets the CSF key under the default map name `oracle.wsm.security` in the credential store. To differentiate between several CSF entries in the credential store, the domain name will be appended to `sign-csf-key` and `enc-csf-key`.

The following examples provide sample commands for configuring the different keystore types:

- **KSS Keystore**

```
wls:/base_domain/serverConfig> configureWSMKeystore('/wls/
new_domain',keystoreType='KSS',
location='kss://owsm/keystore', signAlias='orakey', cryptAlias='orakey')
```

```
Successfully configured property "keystore.type".
Successfully configured property "location".
Successfully configured property "keystore.pass.csf.key".
Successfully configured property "keystore.sig.csf.key".
Successfully configured property "keystore.enc.csf.key".
```

- **JKS Keystore**

```
wls:/base_domain/serverConfig> configureWSMKeystore('/wls/
new_domain',keystoreType='JKS',
location='./default-keystore.jks',
keystorePassword='password',signAlias='orakey',
signAliasPassword='password',cryptAlias='orakey',cryptAliasPassword='password')
```

```
Successfully configured property "keystore.type".
Successfully configured property "location".
Successfully configured property "keystore.pass.csf.key".
Successfully configured property "keystore.sig.csf.key".
Successfully configured property "keystore.enc.csf.key".
```

- **PKCS11 Keystore**

```
wls:/base_domain/serverConfig> configureWSMKeystore('/wls/
new_domain',keystoreType='PKCS11',
keystorePassword='password',signAlias='orakey', signAliasPassword='password',
cryptAlias='orakey',cryptAliasPassword='password')
```

```
Successfully configured property "keystore.type".
Successfully configured property "location".
Successfully configured property "keystore.pass.csf.key".
Successfully configured property "keystore.sig.csf.key".
Successfully configured property "keystore.enc.csf.key".
```

- **HSM Luna SA Keystore**

```
wls:/base_domain/serverConfig> configureWSMKeystore('/wls/
new_domain',keystoreType='LUNA',signAlias='orakey', cryptAlias='orakey')
```

```
Successfully configured property "keystore.type".
Successfully configured property "location".
Successfully configured property "keystore.pass.csf.key".
Successfully configured property "keystore.sig.csf.key".
Successfully configured property "keystore.enc.csf.key".
```

5. If you are configuring the keystore for the first time, a server restart is not required. If, however, you are changing the keystore or keystore configuration, you must restart the server.

14.9.2 Configuring Security Policy Enforcement Using WLST

You can use the `setWSMConfiguration` command to configure security policy enforcement.

The command is described in "[Setting OWSM Domain Configuration Properties Using the setWSMConfiguration Command](#)".

[Table 14-6](#) lists the OWSM domain-level configuration properties that you can set for security policy enforcement.

Table 14-6 Security Policy Enforcement Domain-Level Configuration Properties

Category	Property Name	Default	Description
Agent	<code>allow.all.xpath</code>	<code>false</code>	<p>Specifies whether OWSM will accept all types of XPath transformations. By default, OWSM only accepts the XPath transformation <code>ancestor-or-self::*[namespace-uri()='<namespace>' and local-name()='<name>']</code> inside the signature (in the incoming SOAP message).</p> <p>Set this property to <code>true</code> to allow and accept all types of XPath transformations. Note that enabling this property may result in XPath based Denial of Service attacks or other similar XPath based security vulnerabilities.</p>
Agent	<code>use.unified.fault.code</code>	<code>true</code>	<p>Specifies whether OWSM will send <code>InvalidSecurity</code> fault codes for all types of errors. If a web service sends different fault codes for different types of errors (such as <code>FailedAuthentication</code>, <code>InvalidSecurityToken</code>, <code>FailedCheck</code>, and so on) there is a possibility of an XML encryption attack.</p> <p>Note: This applies only to security-related faults that originated from OWSM. It does not apply to business faults coming from web services.</p> <p>To avoid these types of attacks, OWSM will send only <code>InvalidSecurity</code> fault codes for all types of errors when this property is set to <code>true</code> (the default).</p> <p>Set this property to <code>false</code> if it is required to send different fault codes for the different scenarios. Oracle does not recommend changing this setting to <code>false</code> as it might lead to an XML encryption attack.</p>
Agent	<code>client.clock.skew</code>	<code>0</code>	<p>Tolerance of time, in seconds, that is used to calculate the <code>NotBefore</code> and <code>NotOnOrAfter</code> conditions for SAML token generation. Together, these conditions define the lower and upper boundaries to limit the validity of the token.</p>

Table 14-6 (Cont.) Security Policy Enforcement Domain-Level Configuration Properties

Category	Property Name	Default	Description
Agent	compliance.check	true	This property enables compliance checking for various security related configurations/checks and enforces that the incoming message complies to the security requirement of the receiving party. This setting applies to the request message coming to service and the response message coming to the client. It checks for encryption and signature algorithms, encrypted and signed elements and parts, attachment compliance, timestamp compliance, and so on. Setting this property to <code>false</code> allows you to skip all of these compliance checks, but this is strongly discouraged.
Agent	clock.skew	360000	Tolerance of time differences 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. Note: This property must have a value greater than or equal to 1 second. If the value is less than 1 second, then the property does not work as expected. For more information about this property, see the description for the Clock Skew property in " Configuring Security Policy Enforcement Using Fusion Middleware Control ".
Agent	expire.time	300000	Duration of time 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. For more information about this property, see the description for the Message Expiration Time property in " Configuring Security Policy Enforcement Using Fusion Middleware Control ".
Agent	nonce.ttl	28800000	Total time-to-live 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. For more information about this property, see the description for the Nonce Time To Live property in " Configuring Security Policy Enforcement Using Fusion Middleware Control ".
Agent	signature.cache.enable	false	If enabled, OWSM will cache the primary signature of SOAP messages in the signature cache for a stipulated time. If a message comes in this time with the same signature, it will be rejected as a duplicate message. To get this functionality <code>include-timestamp</code> flag in the policy should be <code>true</code> . If it is not <code>true</code> , OWSM will not know that the coming message is duplicate, and the message will be accepted as genuine. By default this property is disabled. You can enable it in the WSM Domain Configuration page. You can also enable it through WLST.

14.9.3 Configuring Identity Extension Properties Using WLST

You can use the `setWSMConfiguration` command to configure identity settings.

The command is described in "[Setting OWSM Domain Configuration Properties Using the setWSMConfiguration Command](#)".

For more information about the service identity certificate extension and hostname verification, see "[Understanding Service Identity Certificate Extensions](#)".



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 "[Overview of Configuring Keystores for Message Protection](#)".

[Table 14-7](#) lists the OWSM domain-level configuration properties that you can set for service identity.

Table 14-7 Identity Domain-Level Configuration Properties

Category	Property Name	Default	Description
Identity	ignore.hostname.verification	true	Specifies whether to ignore the hostname verification feature per domain. By default this property is disabled (true). However, hostname verification can be enabled by setting the property to false.
Identity	ignore.identity.wsdl	false	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 runtime for encryption. The consumption of the X509 Certificate can be disabled by changing the default setting to true.

14.9.4 Configuring Secure Conversation for the Domain Using WLST

You can use the `setWSMConfiguration` command to configure domain-level secure conversation.

The command is described in "[Setting OWSM Domain Configuration Properties Using the setWSMConfiguration Command](#)".

For more information about domain-level secure conversation, see "[Secure Conversation Configuration for the Domain Using Fusion Middleware Control](#)".

[Table 14-8](#) lists the OWSM domain-level configuration properties that you can set for secure conversation.

Table 14-8 Secure Conversation Domain-Level Configuration Properties

Category	Property Name	Default	Description
SecureConversation	token.lifetime	1800000	The default number of seconds after which the secure conversation session should expire.
SecureConversation	token.lifetime.reauth	28800000	The default number of seconds after which secure conversation session of re-authenticate use cases should expire. The reauthenticate lifetime session should usually have a larger value because the session is shared across multiple users. When reauthentication is true and the session is shared among many users, you can set this to a larger value.

Table 14-8 (Cont.) Secure Conversation Domain-Level Configuration Properties

Category	Property Name	Default	Description
SecureConversation	rm.encrypt.body	false	(Applies to web service client only.) Specifies whether the WS-RM protocol messages should be encrypted. If set to <code>true</code> , the body of protocol request messages such as <code>createSequence()</code> and <code>terminateSequence()</code> are encrypted. The response message body for protocol messages depends on the request message body: if the request message from the client is encrypted for protocol messages, the web service sends the response encrypted, and vice versa.

14.10 About Configuring Policy Access Using WLST

You can use WLST to set domain-level message policy access for OWSM, including the ability to configure the Policy Manager connection for auto-discovery, and whether SSL should be used for the connection. You can also manage the policy cache and configure high availability by tuning the retry logic.

The configuration details are provided in the following sections:

- [Configuring the Policy Manager Connection Using WLST](#)
- [Updating Bootstrap Configuration Properties Using the `setWSMBootstrapConfig` Command](#)
- [About Refreshing Configuration Cache in OWSM Manually by using WLST](#)
- [Configuring High Availability and Cache Management Using WLST](#)

14.10.1 Configuring the Policy Manager Connection Using WLST

You can use the `setWSMConfiguration` command to configure the Policy Manager connection.

The command is described in "[Setting OWSM Domain Configuration Properties Using the `setWSMConfiguration` Command](#)".

For additional information about the Policy Manager connection, see "[Understanding Configuring the Policy Manager Connection Using Fusion Middleware Control](#)".

[Table 14-9](#) lists the OWSM domain-level configuration properties that you can set for the Policy Manager connection.

Table 14-9 Policy Manager Connection Domain-Level Configuration Properties

Category	Property Name	Default	Description
ConfigManager	connect.retry.delay	60000	Number of milliseconds to wait before the Agent attempts to establish a connection to the Policy Manager after a failure.
ConfigManager	keystore.csf.key	keystore-csf-key	CSF key to use to obtain credential of the keystore. Required for two-way SSL.

Table 14-9 (Cont.) Policy Manager Connection Domain-Level Configuration Properties

Category	Property Name	Default	Description
ConfigManager	keystore.path	kss://owsm/ keystore	Keystore path relative to the domain configuration directory. If the keystore path is provided as <code>kss://owsm/keystore</code> , then the absolute path of the keystore is computed as <code>kss://<stripe>/<keystore></code> . If the keystore path is provided as <code>tmp/my.keystore.jks</code> , then the absolute path of the keystore is computed as <code>/domain_config_directory/fmwconfig/tmp/my.keystore.jks</code> . Required for two-way SSL.
ConfigManager	keystore.ssl.alias	mykey	Keystore alias. Required for two-way SSL.
ConfigManager	keystore.type	KSS	Type of keystore. Default value is KSS. Required for two-way SSL.
ConfigManager	pm.csf.key		CSF key to use to obtain a principal (and its credentials) authorized to access an OWSM Policy Manager instance. If not specified, then the OracleSystemUser principal (standard for WebLogic Server environments) is used.
ConfigManager	pm.url		URL that specifies the location of the policy accessor. The following entries are valid: <ul style="list-style-type: none"> <code>file://location_of_mds_home</code> — Repository is accessed directly as an MDS instance (supported only in Java SE) <code>classpath://JAR_location</code> — JAR file containing the predefined policies and assertion templates is accessed using the classpath location Note: Because classpath mode is a read-only Policy Manager mode, only design time policy attachments will be in effect. No post-deployment policy attachments or global policy sets will be in effect in this mode. <code>auto</code> — Enables auto-discovery of the Policy Manager using non-secure protocol. If the auto-discovery logic cannot connect to a Policy Manager using non-secure protocol because the non-secure port is disabled, it will attempt to connect to a Policy Manager using secure protocol. <code>auto-ssl</code> — Enables auto-discovery of the Policy Manager using secure protocol. If SSL is enabled, then the auto-discovery logic will only connect to Policy Managers using secure protocol and will not try to connect to any Policy Managers deployed on non-SSL servers, even if the SSL-enabled server goes down. Note: If you do not provide a value for <code>pm.url</code>, OWSM uses auto-discovery in the same domain using non-secure protocol. If the auto-discovery logic cannot connect to a Policy Manager using non-secure protocol because the non-secure port is disabled, it will attempt to connect to a Policy Manager using secure protocol.
ConfigManager	refresh.repeat	600000	Number of milliseconds to wait between configuration refreshes.

Table 14-9 (Cont.) Policy Manager Connection Domain-Level Configuration Properties

Category	Property Name	Default	Description
ConfigManager	ssl.twoway	false	Specifies if SSL connection should be one-way or two-way. Set to <code>true</code> for two-way SSL.
ConfigManager	truststore.csf.key	keystore-csf-key	CSF key to use to obtain credential of the keystore. Required for one-way and two-way SSL.
ConfigManager	truststore.path	kss://owsm/keystore	Truststore path relative to the domain configuration directory. If the truststore path is provided as <code>kss://owsm/keystore</code> , then the absolute path to the truststore is computed as <code>kss://<stripe>/<keystore></code> . If the truststore path is provided as <code>tmp/my.truststore.jks</code> , then the absolute path to the truststore is computed as <code>/domain_config_directory/fmwconfig/tmp/my.truststore.jks</code> . Required for one-way and two-way SSL.

14.10.2 Updating Bootstrap Configuration Properties Using the `setWSMBootstrapConfig` Command

You can use the `setWSMBootstrapConfig` command to update the bootstrap configuration document (`domain_home/config/fmwconfig/wsm-config.xml`).

To do so, complete the following steps:

1. Connect to the running instance of the server in the domain to be configured. See *Accessing the Web Services Custom WLST Commands* in *Administering Web Services*.
2. Run the `setWSMBootstrapConfig` command to update the desired configuration properties.

```
setWSMBootstrapConfig (domainName, domainHome, propertyCategory, propertyName,
propertyValue, [raiseError='true|false'])
```

In this command:

- `domainName`—Specify the domain name.
- `domainHome`—Specify the root directory of the domain.
- `propertycategory`—The category of the property.
- `propertyname`—The name of the property. Specify one of the following supported property:
 - `pm.url`: URL that specifies the location of the OWSM Policy Manager instance. Supported `pm.url` protocol are as follows:
 - File - This is supported only in Java SE. The OWSM repository will be accessed directly as a Plain Old Java Object (POJO) or OWSM Policy Manager.
 - http/https - If the URL protocol is http or https, then the repository will be accessed through the OWSM Policy Manager's RESTful APIs.

 **Note:**

If the URL protocol is https then you must configure the following properties:

- * truststore.path
- * truststore.csf.key
- * ssl.twoway
- * keystore.path
- * keystore.ssl.alias
- * keystore.type
- * keystore.ssl.alias

t3/t3s - If the URL protocol is t3 or t3s then the repository will be accessed through the OWSM Policy Manager's EJB APIs.

 **Note:**

If the `pm.url` is not configured explicitly, then by default OWSM repository is accessed as OWSM Policy Manager's EJB APIs. This is recommended approach for WebLogic domain.

- `pm.csf-key`: CSF key to use to obtain a principal (and its credentials) authorized to access an OWSM Policy Manager instance.
- `propertyvalues`—The values to set for a property in the configuration document.
- `raiseError` - Optional. When set to `true`, it raises exception in case of known errors. When set to `false`, it returns a boolean `false` value in case of known errors. By default, it's set to `true`.

Example:

```
setWSMBootstrapConfig ('base_domain', '/ORACLE_HOME/domains/base_domain',
'ConfigManager', 'pm.url', ['t3://localhost:7001'])
```

14.10.3 About Refreshing Configuration Cache in OWSM Manually by using WLST

OWSM has background threads to refresh configuration and documents cache automatically at regular time intervals. This interval is specified in the following properties: **refresh.repeat** and **cache.refresh.repeat**. You can disable this automatic refresh by setting the value of configuration property **auto.refresh** to disabled. Then, you can invoke the WLST command **refreshWSMCache()** to refresh this cache manually when required.

The following topics explain this further:

- [Disabling Automatic Refresh Option in OWSM by using WLST](#)
- [Checking Status of Automatic Refresh in OWSM by using WLST](#)
- [Refreshing the OWSM Cache Manually by using WLST](#)

14.10.3.1 Disabling Automatic Refresh Option in OWSM by using WLST

You can disable the automatic refresh of OWSM cache by setting the value of the configuration property `auto.refresh` to `disabled`.

To disable the automatic refresh of OWSM cache:

1. Connect to the running instance of the server in the domain to be configured as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Disable the OWSM auto-refresh threads by running the following command:

```
WLST> setWSMConfiguration('<context>', 'ConfigManager', 'auto.refresh', None, ['disabled'])
```

Example:

```
WLST> setWSMConfiguration(None, 'ConfigManager', 'auto.refresh', None, ['disabled'])
WLST> setWSMConfiguration('/WLS/myDomain', 'ConfigManager', 'auto.refresh', None, ['disabled'])
```

14.10.3.2 Checking Status of Automatic Refresh in OWSM by using WLST

You can check the status of automatic refresh setting of OWSM cache by using the WLST command `displayWSMConfiguration()`.

To check the status of automatic refresh:

1. Connect to the running instance of the server in the domain to be configured as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Run the following command:

```
WLST> displayWSMConfiguration()
```

Example:

```
WLST> displayWSMConfiguration()
WLST> displayWSMConfiguration('/WLS/myDomain')
```

In the output look for the value of `auto.refresh` as `disabled` with the following syntax:

```
NAME: "auto.refresh" CATEGORY: "ConfigManager" SOURCE: "<context>" Value: <value>
```

14.10.3.3 Refreshing the OWSM Cache Manually by using WLST

You can invoke the WLST command `refreshWSMCache()` to refresh this cache manually when required.

To refresh the OWSM cache manually:

1. Connect to the running instance of the server in the domain to be configured as described in "Accessing the Web Services Custom WLST Commands" in *Administering Web Services*.
2. Run the `refreshWSMCache()` command.

This WLST updates the documents and configuration cached by OWSM.

14.10.4 Configuring High Availability and Cache Management Using WLST

You can use the `setWSMConfiguration` command to configure high availability and cache management.

The command is described in "[Setting OWSM Domain Configuration Properties Using the `setWSMConfiguration` Command](#)".

For more information about tuning the connection using these properties, see "[High Availability Configuration and Cache Management Using Fusion Middleware Control](#)".

[Table 14-10](#) lists the OWSM domain-level configuration properties that you can set for high availability and cache management.

Table 14-10 High Availability and Cache Management Domain Configuration Properties

Category	Property Name	Default	Description
BeanAccessor	cache.refresh.initial	600000	The number of milliseconds to wait before initial cache refresh. Note: This property must have a value greater than or equal to 5 second. If the value is less than 5 second, then the property does not work as expected.
BeanAccessor	cache.refresh.repeat	600000	The number of milliseconds to wait between cache refreshes Note: This property must have a value greater than or equal to 5 second. If the value is less than 5 second, then the property does not work as expected. A lower value will also overload the system.
BeanAccessor	failure.retry.count	2	The number of times to retry after a communication failure.
BeanAccessor	failure.retry.delay	5000	The number of milliseconds to wait between retry attempts.
BeanAccessor	jndi.connection.timeout	30000	The number of milliseconds to wait before timing out for waiting for the connection to the configured Policy Manager using the <code>java.naming.provider.url</code> .
BeanAccessor	missing.retry.delay	15000	The number of milliseconds to wait before trying to retrieve a missing document.
BeanAccessor	usage.record.delay	30000	The number of milliseconds to wait before sending usage data.

Managing the Oracle Web Services Manager Repository

The OWSM repository allows you to import, export, backup, and restore Oracle Web Services Manager metadata, such as policies, assertion templates, and policy usage data. Policies in the repository can also be patched and upgraded.

Topics:

- [Overview of OWSM Repository](#)
- [Registering an OWSM Repository](#)
- [Understanding the Different Mechanisms for Importing and Exporting Policies](#)
- [About Importing and Exporting Documents in the Repository Using WLST](#)
- [Exporting Policies from the OWSM Repository for Use in JDeveloper](#)
- [About Patching Policies in the Repository](#)
- [Creating Back Up and Restoring the OWSM Repository](#)
- [Upgrading the OWSM Repository](#)
- [Rebuilding the OWSM Repository](#)
- [Configuring Multiple Domains for a Single OWSM Repository Instance](#)

15.1 Overview of OWSM Repository

Oracle Web Services Manager (WSM) uses an MDS repository to store OWSM metadata, such as policies, assertion templates, and policy usage data. The OWSM 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 OWSM Repository, each policy has a URI that is evaluated to form a path in which to locate a particular XML document containing the policy. OWSM 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, OWSM only accesses the latest version during policy enforcement.

Details about managing the MDS repository are provided in "Managing the MDS Repository" in *Administering Oracle Fusion Middleware*.

15.2 Registering an OWSM Repository

Before you can deploy an application to an MDS Repository, such as the OWSM Repository, you must register the repository with the Oracle WebLogic domain.

To register an OWSM Repository:

1. In the Navigator pane, expand **Metadata Repositories** and select **mds-owsm**, as shown in [Figure 15-1](#).

Figure 15-1 Metadata Repository in Navigation Pane



2. Select **Metadata Repository**, then **Administration**, then **Register/Deregister**. The Metadata Repositories page is displayed, as shown in [Figure 15-2](#).

Figure 15-2 Registering an OWSM 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 *Administering Oracle Fusion Middleware*.

15.3 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 OWSM Repository.

Fusion Middleware Control provides the ability to selectively import and export one or more user-created policies or assertion templates to/from a zip archive file. Read only documents, including predefined policies and assertion templates, cannot be imported or exported because the same documents would be present in the target environment. The procedures for importing and exporting policies and assertion templates using Fusion Middleware Control are described in the following sections:

- "Importing Web Service Policies"

- ["Exporting Web Service Policies"](#)
- ["Importing an Assertion Template"](#)
- ["Exporting an Assertion Template"](#)

The WLST commands, `importWSMArchive` and `exportWSMRepository`, are provided to facilitate importing and exporting multiple OWSM documents directly to and from the OWSM Repository. For details about using these commands, see ["About Importing and Exporting Documents in the Repository Using WLST"](#).

When you import or export policies using either of these mechanisms, the operation is routed through an instance of the OWSM 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.

15.4 About Importing and Exporting Documents in the Repository Using WLST

You can import and export OWSM documents and application metadata to and from the OWSM Repository using the `importWSMArchive`, `exportWSMRepository`, and `exportWSMAppMetadata` commands.

This is described in the following sections:

- [Exporting Documents from the Repository Using WLST](#)
- [Exporting Application Metadata from the Repository Using WLST](#)
- [Importing Documents into the Repository Using WLST](#)

For more information about the WLST commands and their arguments, see "Web Services Custom WLST Commands" in *WLST Command Reference for Infrastructure Components*.

15.4.1 Exporting Documents from the Repository Using WLST

To export documents from the repository to a supported ZIP archive file, use the `exportWSMRepository` command.

```
exportWSMRepository(archive, [documents=None], [includeShared='false'])
```

Note the following:

- Read only documents, including predefined policies and assertion templates, will not be exported because the same documents already exist in the target environment.
- 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 no documents are specified, then only shared documents that include policies and policy sets are exported. Because read-only documents cannot be exported, only custom or cloned shared policies will be included in the export. If this argument is specified as an empty string [''], then all shared documents that include policies and policy sets, application metadata and configuration documents 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 user-created policies whose URI begins with either "test/wss_" or "test/wss11_", enter the following:

```
wls:/jrfServer_domain/serverConfig>exportWSMRepository('/tmp/test2.zip',
['policies:test/wss_%','policies:test/wss11_%'])

Exporting "/policies/test/
wss11_x509_token_with_message_protection_service_policy_test"
Exporting "/policies/test/wss_username_token_over_ssl_service_policy_Test"
Successfully exported "2" documents.
```

- Use the optional `includeShared` argument to include the shared documents (those that are specified as policy references within policy sets and wsm-assembly documents) as part of same archive during the export. Because read-only documents can not be exported, only custom or cloned shared policies will be included in the export. The default is `false`.

For example, to export active policy set documents and the policies they use:

```
wls:/jrfServer_domain/serverConfig>exportWSMRepository('/tmp/repository-active.jar',
['policysets:global/%'], true)

Exporting "/policies/test/wss_username_token_over_ssl_service_policy_Test"
Exporting "/policysets/global/all-domains-default-web-service-policies"
Exporting "/policysets/global/app-only-web-service-policies"
Exporting "/policysets/global/migrate_example"
Successfully exported "4" 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>exportWSMRepository('/tmp/repository-backup.jar',
['/policysets/global/module-web-service-policies'])
```

15.4.2 Exporting Application Metadata from the Repository Using WLST

To export application metadata from the repository to a supported ZIP archive file, use the `exportWSMAppMetadata` command.

```
exportWSMAppMetadata (archive, [applications=None], [includeShared='false'])
```

Note the following:

- This command is supported for Oracle Infrastructure and RESTful web services only. This command is not supported for ADF DC web service clients and Java EE web services.
- 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. If you choose the `overwrite` option, the original archive is backed up and a message describes the location of the backup archive.
- Use the optional `applications` argument to specify the applications for which you want to export the metadata to the archive. If no application names are specified, then the metadata for all the applications in the domain is exported. You can specify a list of search expressions to find specific application metadata in the repository, using this syntax: `/ {PLATFORM_NAME}/{DOMAIN_NAME}/{APPLICATION_NAME}`.
- Use the optional `includeShared` argument to specify whether the shared documents (those that are specified as policy references within wsm-assembly documents) should be included in the export. The default is `false`. Because read-only documents can not be exported, only custom or cloned shared policies will be included in the export.

For example, the following command exports the application metadata for an application whose name begins with `jaxws` into the `applications.zip` file in the `tmp` directory:

```
wls:/jrfServer_domain/serverConfig>exportWSMRepository('/tmp/applications.zip',['/WLS/  
base_domain/jaxws%'])
```

```
Exporting "\assembly\WLS\base_domain\jaxwsejb30ws\jaxwsejb\wsm-assembly.xml"Successfully  
exported "1" documents.
```

15.4.3 Importing Documents into the Repository Using WLST

To import documents into the repository, use the `importWSMArchive` command.

```
importWSMArchive(archive,[map=none],[generateMapFile='false'])
```

Note the following:

- Read only documents, including predefined policies and assertion templates, will not be imported because the same documents already exist in the target environment.
- The `archive` argument, which is required, specifies the path to the archive file that contains the list of documents to be imported.
- 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 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>importWSMArchive('/tmp/repository-active.jar',  
'/tmp/mapfile.txt', true)
```

```
Successfully generated "Documents 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>importWSMArchive('/tmp/repository-active.jar', '/tmp/
mapfile.txt')

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 "3" documents
```

15.5 Exporting Policies from the OWSM Repository for Use in JDeveloper

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 `importWSMArchive` and `exportWSMRepository` commands for policy migration.

This is described in "Migrating Policies" in *Administering Web Services*.

In JDeveloper, you can add custom policies to the default policy store location at:

```
JDEV_USER_HOME\system12.1.2.0.x.x.x\DefaultDomain\store\gmds\owsm
```

If not set, `JDEV_USER_HOME` defaults to `C:\Users\user-dir\AppData\Roaming\JDeveloper`.

In this default policy store location, OWSM policies must be included using the directory structure `policies/policyname`.



Note:

Within this directory structure, `policyname` includes the directory in which the policy is located. For example, all predefined policies provided by Oracle are contained in the `oracle/` directory, such as `oracle/wss_http_token_service_policy`.

Oracle recommends that custom policies be located in a directory that is separate from the `/oracle` directory that contains the predefined policies, for example `mycompany/mypolicy`.

When exporting policy files from the OWSM Repository using WLST or Fusion Middleware Control, they are exported into a zip archive using the following directory structure:

```
META-INF/policies/policyname
```

To use the policies in a JDeveloper environment, you must extract the contents of the `META-INF` directory in the zip archive into the default JDeveloper policy store location noted above. When you do so, the `policies/policyname` structure is maintained and the policies will be available in the JDeveloper environment. If you do not maintain this directory structure, the policies will not be available in the JDeveloper environment.

15.6 About Patching Policies in the Repository

You can patch the OWSM Repository using either Fusion Middleware Control or the WLST commands.

This is described in "[Understanding the Different Mechanisms for Importing and Exporting Policies](#)".

When you create or update a policy, there are two possible scenarios to consider when you patch the repository:

 **Note:**

Predefined policies are read-only and cannot be updated or overwritten.

- You create a new policy or update an existing user-defined 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 user-defined 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.

15.7 Creating Back Up and Restoring the OWSM Repository

Use the `exportWSMRepository` and `importWSMArchive` WLST commands to back up and restore the OWSM Repository.

For more information about these commands, see "[About Importing and Exporting Documents in the Repository Using WLST](#)".

For example, to backup all the OWSM artifacts in the repository, enter the following command:

```
wls:/jrfServer_domain/serverConfig>exportWSMRepository ('/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 `importWSMArchive` command to import all the OWSM 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>importWSMArchive ('/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 *WLST Command Reference for Infrastructure Components*.

15.8 Upgrading the OWSM Repository

You can upgrade the OWSM repository from 11g to 14c.

The predefined OWSM documents distributed in the 14c release, including predefined policies and assertion templates, are now read only. During the upgrade from 11g to 14c, the following occurs:

- Existing predefined documents that have not been modified will be replaced with the 14c read-only versions.
- New predefined documents introduced in 14c will be added.
- Existing predefined documents that have been modified, and any user-created documents, will not be overwritten and remain unchanged.

To ensure that you always have the latest versions of the Oracle predefined documents, Oracle recommends that you do the following:

1. Clone any of the modified documents as described in the following topics:
 - ["Cloning a Web Service Policy"](#)
 - ["Cloning an Assertion Template"](#)
2. Replace any policy references to the modified documents with policy references to the cloned versions.
3. Upgrade the OWSM repository using the `resetWSMRepository(clearStore='false')` command. Using the `clearStore='false'` option deletes and replaces only the predefined documents provided by Oracle. Custom documents and predefined documents that you cloned are not deleted. This is the default setting for this command.

Note:

The `resetWSMRepository` command also updates the version number of the predefined policies and assertion templates.

To delete all documents in the repository, including custom documents, see ["Rebuilding the OWSM Repository"](#).

15.9 Rebuilding the OWSM Repository

In some circumstances, it may be desirable to delete all of the documents in the OWSM Repository and replace them with the latest set that was provided with your Fusion Middleware installation. For example, when starting a new project in a test environment it may be useful to reset the repository contents to their original state.

 **Note:**

The procedure described in this section deletes all documents in the OWSM Repository, including cloned and user-created documents. To delete and replace only the predefined documents provided by Oracle, see "[Upgrading the OWSM Repository](#)".

To rebuild the OWSM 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" in *Administering Web Services*.

 **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 `resetWSMRepository(clearStore='true')` command to delete all the documents, including custom user documents, from the OWSM Repository and repopulate it with full set of predefined read-only documents that were provided by Oracle.

For more information about the `resetWSMRepository` WLST command, see "OWSM Repository Management Commands" in *WLST Command Reference for Infrastructure Components*.

 **Note:**

Before you delete a policy, Oracle recommends that you verify that the policy is not attached to any policy subjects.

15.10 Configuring Multiple Domains for a Single OWSM Repository Instance

You can use a centralized OWSM Repository to import, export, backup, and restore Oracle Web Services Manager metadata, such as policies, assertion templates, and policy usage data

for multiple domains. Each domain can have its own OWSM Policy Manager wired to a shared OWSM Repository.

The following OWSM documents and Application Metadata can be shared across multiple shared domains using a single OWSM repository instance:

- Policies & Assertions
- Global Policy Sets
- Token Issuer Trust

 **Note:**

Any change made to the default configuration document in a domain cannot be shared across multiple domain, because customizing configuration document is domain specific.

To configure the OWSM repository for multiple domain, do the following:

Prerequisites

Before configuring the domains for a single OWSM repository instance, you must complete the following:

- You must install the Metadata Services (MDS), Oracle Platform Security Services (OPSS), Service Table (STB), and Audit Services Append (IAU_APPEND) schemas using the Repository Creation Utility (RCU). See *Creating Database Schemas in Oracle Fusion Middleware Installing and Configuring the Oracle Fusion Middleware Infrastructure*.

 **Note:**

- The database schema associated with the domain being created must be brand new and cannot be one that has been previously used.
 - You must create the schemas for each domain that will be managed by a centralized repository.
- If your domains are sharing a single database instance then you must associate the domains with the database, as described in *Sharing a Database Instance in Oracle Fusion Middleware Securing Applications with Oracle Platform Security Services*
 - Configure the OWSM domains using the Configuration Wizard, as described in *Configuring Your WebLogic Domain in Oracle Fusion Middleware Installing and Configuring the Oracle Fusion Middleware Infrastructure*.

 **Note:**

From the Administration Server screen, ensure that you assign a user-expandable server group to the Administration Server from the **Server Group** drop-down list.

- Verify the service table entries and components using the the Fusion Middleware Control, as described in [Verifying Service Table Entries and Components Using Fusion Middleware Control](#).
- Verify that the OWSM Agent is wired correctly to the appropriate Policy Manager URL in a domain, as described in [Verifying Agent Bindings Using Fusion Middleware Control](#).

Configuring the Domains

To share the OWSM documents and Application Metadata across multiple shared domains, complete the following:

1. Create a global policy set to secure the domains, by running the `createWSMPolicySet` command:

```
createWSMPolicySet(name, type, attachTo, [description=None], [enable='true'])
```

See Also:

[Creating a New Policy Set Using WLST](#)

2. Create a token issuer trust domain document for the domains (For example, `base_domain1` and `base_domain2`) sharing a single OWSM Repository instance, using the `createWSMTokenIssuerTrustDocument` command.

- a. Create and configure token issuer trust domain document for the `base_domain1` by running `createWSMTokenIssuerTrustDocument` command:

```
wls:/base_domain/serverConfig>
createWSMTokenIssuerTrustDocument('tokenissuertrustWLSbase_domain1')
New Token Issuer Trust document named "tokenissuertrustWLSbase_domain1"
created.
Run the setWSMConfiguration command where category = "TokenIssuerTrust",
property name = "name" and value = "tokenissuertrustWLSbase_domain", for the
new
document to be used in the domain configuration.
wls:/base_domain/serverConfig> setWSMConfiguration
(None, 'TokenIssuerTrust', 'name', None, ['tokenissuertrustWLSbase_domain1'])
The values "[tokenissuertrustWLSbase_domain]" have been added to property
"name" within category "TokenIssuerTrust".
Configuration properties associated with the context "/WLS/base_domain1" has
been updated.
```

- b. Configure token issuer trust domain document for the `base_domain2` by running `createWSMTokenIssuerTrustDocument` command:

```
wls:/base_domain/serverConfig> setWSMConfiguration
(None, 'TokenIssuerTrust', 'name', None, ['tokenissuertrustWLSbase_domain2'])
The values "[tokenissuertrustWLSbase_domain]" have been added to property
"name" within category "TokenIssuerTrust".
Configuration properties associated with the context "/WLS/base_domain" has
been updated.
```

3. Create the OPSS Keystore to manage keys and certificates for SSL, message security, encryption, and related tasks. After you create the keystore, you must configure it for the domains (For example, `base_domain1` and `base_domain2`) by running the `configureWSMKeystore` command:

```
configureWSMKeystore(context, keystoreType, location, keystorePassword, signAlias,
signAliasPassword, cryptAlias, cryptAliasPassword)
```

 **See Also:**

[Configuring the OWSM Keystore Using WLST](#)

16

Diagnosing Problems with Oracle Web Services Manager

You can diagnose and fix common problems with Oracle Web Services Manager (OWSM), such as security problems with keys and credentials, certificates, policy access, assertions, and so on. You can also use WLST to diagnose problems.

Topics:

- [Diagnosing Policy Manager Problems Using the OWSM Policy Manager Page](#)
- [Overview of Common Problems with Oracle Web Services Manager](#)
- [Overview of Policy Attachment Issues Using WLST](#)
- [About Diagnosing Problems With a Domain Configuration Using WLST](#)
- [Common Oracle Web Services Manager Exceptions for WS-Trust Use Cases](#)
- [Managing third party server integration with OWSM](#)

16.1 Diagnosing Policy Manager Problems Using the OWSM Policy Manager Page

The OWSM Policy Manager manages all OWSM policies and needs to be running to use the OWSM policy framework. You can check the current state of the Policy Manager and review its response time, load, and other data from the OWSM Policy Manager page in Oracle Enterprise Manager Fusion Middleware Control.

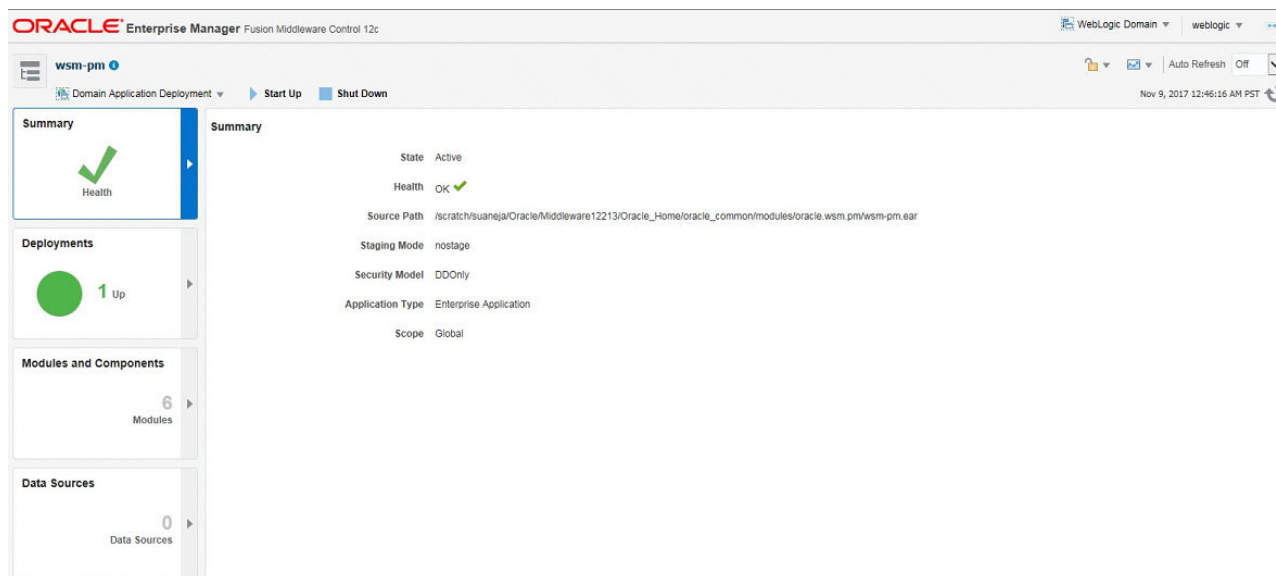
If the OWSM Policy Manager is not available, then any modifications that you make to policy attachments for ADF and WebCenter services and clients will not be saved to the OWSM repository.

To view the OWSM Policy Manager page:

1. In the Target Navigation pane, expand **Application Deployments**.
2. Under Application Deployments, expand **Internal Applications**.
3. Expand **wsm-pm** and select the **wsm-pm** application.

The OWSM Policy Manager home page is displayed.

Figure 16-1 OWSM Policy Manager Page



From the Policy Manager page, you can perform one or more of the following tasks:

- In the Summary section of the page, you can check the current state of the Policy Manager and identify the server to which it is deployed.
- In the Deployments section of the page, you can view the Name, Status, State, Health, Type and Targets of the deployments.
- In the Modules and Components section of the page, you can check the modules, components, test points and web services deployed. You can validate the connection to the Policy Manager by clicking the Test Point URL for wsm-pm. On the Validate Policy Manager page, click **Validate Policy Manager**.
- In the Data Sources section of the page, you can check the data sources if any and their location.

 **Note:**

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 OWSM policies in the OWSM repository
- The name, latest version, and description of all the OWSM policies in the OWSM repository.
- The total number of OWSM assertion templates in the repository

- The name, latest version, and description of all the OWSM assertion templates in the OWSM repository.
- The creation date and build label of the repository.

For details about the OWSM repository, see [Managing the Oracle Web Services Manager Repository](#) .

16.2 Overview of Common Problems with Oracle Web Services Manager

You may encounter some common problems while using OWSM. This section discusses those problems, as well as possible solutions.

Topics:

- [Overview of Common Policy Manager Connection Problems](#)
- [Overview of Key Store or Credential Store Errors After an Application Invokes a Web Service](#)
- [Overview of Trust Certificate Error After Application Invokes a Web Service](#)
- [About Troubleshooting SAML Assertion Errors During Identity Propagation](#)
- [Overview of Policy Access Problems After an Application Invokes a Web Service](#)
- [Overview of Problems Accessing Users in the Credential Store](#)
- [Overview of Common User Authorization Problems After an Application Invokes a Web Service](#)
- [Overview of Timestamp Errors After an Application Invokes a Web Service](#)
- [Overview of Multiple Authentication Security Policy Errors After an Application Invokes a Web Service](#)

16.2.1 Overview of Common Policy Manager Connection Problems

You might encounter some common Policy Manager connection issues. This section helps you diagnose and resolve those problems.

It includes the following topics:

- [Understanding Common Policy Manager Connection Problems](#)
- [Solving Policy Manager Connection Problems](#)

16.2.1.1 Understanding Common Policy Manager Connection Problems

When a connection to the Policy Manager fails, one or more of the following error messages are displayed:

- 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.
- WSM-02120: Unable to connect to the policy access service.

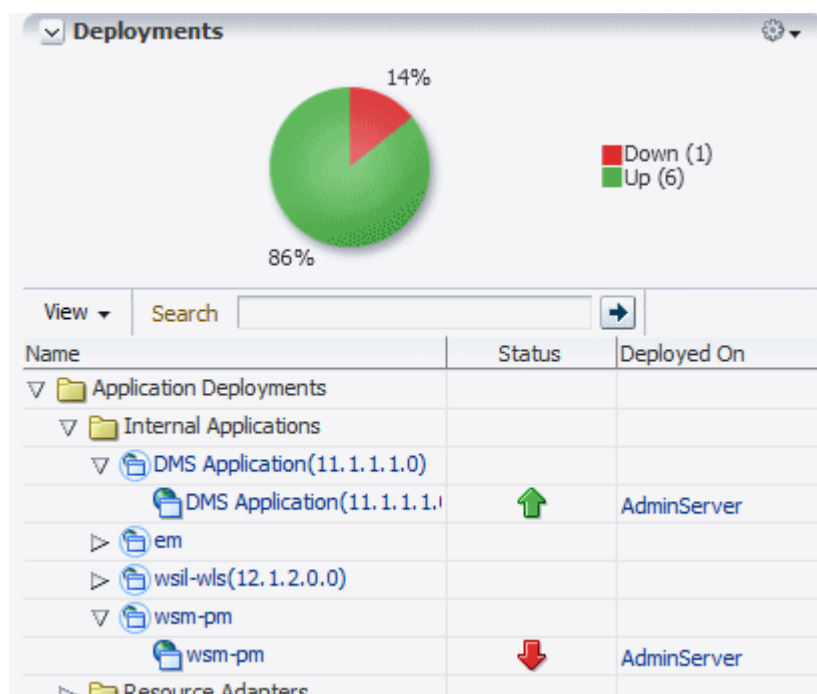
Problems connecting to the Policy Manager are commonly caused by the following:

- 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 OWSM Policy Manager home page, as described in "[Diagnosing Policy Manager Problems Using the OWSM Policy Manager Page](#)" is shown as *Shutdown*.
- The status of the `wsm-pm` internal application on the Domain home page in Fusion Middleware Control is *Down*, as shown in [Figure 16-2](#). To access the Domain home page, select **Home** from the **WebLogic Domain** menu in the top left-hand corner of the page.

Figure 16-2 OWSM Policy Manager Shutdown (Farm Page)



- An error dialog box displays when you attempt to access the OWSM policy management pages in Fusion Middleware Control. This error information is also written to the diagnostic log file, as described in "Reviewing Sample Logs" in *Administering Web Services*.
- The domain is configured to use auto-discovery with SSL.
OWSM supports an auto-discovery feature that it uses to locate and connect to the OWSM Policy Manager. If the domain is configured to use auto-discovery with SSL, then the auto-discovery logic will always try to connect to an SSL-enabled server. 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.
- The cross-component wiring for the Policy Manager and Agent may not be current.

16.2.1.2 Solving Policy Manager Connection Problems

You can use the following information to fix the most common Policy Manager connection problems:

- If the Policy Manager is down, then restart the wsm-pm application as described in "Starting and Stopping Applications" in *Administering Oracle Fusion Middleware*.
- If the domain is configured to use auto-discovery with SSL, then
 - Verify that the wsm-pm Policy Manager application is targeted to an SSL server. Use the WebLogic Server Remote Console as described in "Target an Enterprise application to a server" in the *Oracle WebLogic Server Administration Console Online Help*.
 - Verify that SSL has been configured correctly and that there are no SSL certificate issues. For additional information, see [About Configuring Keystores for SSL](#)
 - If the SSL-enabled server is down, restart the server and the Policy Manager application, as described in "Starting and Stopping Oracle Fusion Middleware" in *Administering Oracle Fusion Middleware*.
 - Verify that the service table and component configuration are correctly configured for SSL. For more information, see [Cross-Component Wiring for Auto-Discovery of Policy Manager](#)
 - If you want to use the Policy Manager on a non-SSL enabled server, then modify the auto-discovery configuration. For more information, see "[Understanding Configuring the Policy Manager Connection Using Fusion Middleware Control](#)" and [Configuring the Policy Manager Connection Using WLST](#)
- If there is a credential issue when attempting to access the Policy Manager, then check the user account. By default, the OWSM run time uses the OracleSystemUser account. If you are not using the default user accounts, you need to modify the configuration as described in [About Modifying the Default User](#)
- If there is a problem with the repository configuration, then
 - Verify that the database and MDS schema are setup correctly. This configuration is performed as part of the installation process. For more information, see "Creating the Database Schemas" in *Installing and Configuring the Oracle Fusion Middleware Infrastructure*.
 - Verify that the JDBC configuration is correct. The JDBC configuration is defined when you create the domain using the Fusion Middleware Configuration Wizard. For more information, see "Configuring Your WebLogic Domain" in *Installing and Configuring the Oracle Fusion Middleware Infrastructure*.
- If there is a problem with the cross-component wiring, then
 - Verify that the service table entry corresponding to the OWSM Policy Manager contains the correct URL.
 - Verify that the wiring between the OWSM Agent Hook and the OWSM Policy Manager is in the `wired` state.

For more information, see the following topics:

- [Verifying Service Table Entries and Components Using Fusion Middleware Control](#)

- [Verifying Agent Bindings Using Fusion Middleware Control](#)

16.2.2 Overview of Key Store or Credential Store Errors After an Application Invokes a Web Service

You may need information to diagnose and resolve common problems that cause key store or credential store errors after an application invokes a web service.

Topics:

- [Understanding Common Key or Credential Store Errors](#)
- [Solving Key and Credential Store Problems](#)

16.2.2.1 Understanding Common Key or Credential Store Errors

If a key store or credential store problem occurs after an application invokes a web service, then an error message similar to the following is displayed:

- WSM-00056: The key `<alias_name>` is not retrieved
- WSM-00256: The property "Keystore Sign Alias" is not set
- WSM-0024: Unable to read key from KSS keystore
- WSM-00340: Key does not exist in the KSS keystore

Key store and credential store errors are commonly caused by the following problems:

- For the JKS keystore:
 - The alias for the signature key or encryption key in the OWSM keystore configuration does not exist in the OWSM keystore.
 - The signature key, encryption key, or OWSM keystore password is not synchronized between the keystore file and the keystore configuration for OWSM. That is, at least one of the passwords does not have identical values in both locations.
 - The signature key is not set.
 - There is a missing key in the credential store.
- For the KSS keystore:
 - The keystore may have not been initialized or the necessary permissions are not granted.
 - The key does not exist in the keystore. The key you entered on the domain configuration Message Security screen cannot be found in the keystore.

16.2.2.2 Solving Key and Credential Store Problems

You can use the following information to fix common key store and credential store problems.

- Verify that the alias for the signature key and encryption key in the OWSM keystore configuration exists in the OWSM keystore file, as follows:
 1. Use Fusion Middleware Control to identify the alias for the signature key and encryption key in the OWSM keystore configuration by performing the procedure in [Configuring the OWSM Keystore](#)
 2. Verify that the aliases identified in step 1 exist in the OWSM keystore file.

For a JKS keystore:

Use the `keytool -list` command on the OWSM keystore file, as described in step 4 of [Generating Private Keys and Creating the Java Keystore](#). 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 OWSM keystore configuration in the credential store. If they are not, you can edit the alias in the OWSM keystore configuration by performing the procedure described in [Configuring the OWSM Keystore](#). You can edit the alias in the OWSM 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 OWSM keystore file, add it as described in [Generating Private Keys and Creating the Java Keystore](#).

For a KSS keystore:

Use Fusion Middleware Control to manage the contents of the keystore and ensure that the alias matches. For more information, see "Managing Keys and Certificates with the Keystore Service" in *Securing Applications with Oracle Platform Security Services*.

- Ensure the passwords are synchronized, as follows:

For a JKS keystore, ensure that the signature key, encryption key, and OWSM keystore file passwords are each synchronized in the keystore file and the keystore configuration for OWSM, as follows:

1. Use `keytool` to reset the passwords in the OWSM 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 OWSM 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 OWSM keystore configuration to the same respective values you set in step 1, as described in [Configuring the OWSM Keystore](#)

16.2.3 Overview of Trust Certificate Error After Application Invokes a Web Service

You may need information to help you diagnose and resolve a trust certificate error that occurs after an application invokes a web service.

Topics:

- [Understanding a Trust Certificate Error](#)
- [Solving Trust Certificate Problems](#)

16.2.3.1 Understanding a Trust Certificate Error

The trust certificate error commonly occurs when the web service is advertising its certificate in the Web Services Description Language (WSDL) and the client may not be configured correctly to trust that certificate or its issuer.

If a trust certificate problem occurs after an application invokes a web service, the following error message is displayed:

```
WSM-00138: The path to the certificate is invalid due to exception
```

16.2.3.2 Solving Trust Certificate Problems

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.

For a KSS keystore, use Fusion Middleware Control to identify the certificates in the client keystore. For more information, see "Managing Keys and Certificates with the Keystore Service" in *Securing Applications with Oracle Platform Security Services*.

For the JKS keystore, 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 OWSM keystore file.

For more information, see [Overview of Policy Configuration Overrides](#)

16.2.4 About Troubleshooting SAML Assertion Errors During Identity Propagation

You may need information to help you troubleshoot SAML assertion problems that occur during identity propagation.

Topics:

- [Understanding Common SAML Assertion Problems](#)
- [Troubleshooting SAML Assertion Problems](#)

16.2.4.1 Understanding Common SAML Assertion Problems

If a problem occurs when an application attempts to propagate a user's identity by calling a different application, then you will see `InvalidSecurityToken`, `FailedAuthentication`, and `SAML assertion issuer` related errors.

SAML assertion errors are commonly caused by the following problems:

- The SAML issuer name for the SAML token is not configured or is configured incorrectly.
- The `subject.precedence` configuration override is set incorrectly.

16.2.4.2 Troubleshooting SAML Assertion Problems

You can use the following information to troubleshoot the most common Policy Manager connection problems:

- 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 [SAML Trusted Issuers and DN Lists Using Fusion Middleware Control](#)

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 [SAML Trusted Issuers and DN Lists Using Fusion Middleware Control](#)
- 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 [Adding Keys and User Credentials to Configure the Credential Store](#)
 3. Ensure the appropriate Web Services Identity Permission is set for the client application by performing the steps in [About SAML Web Service Client Configuration for Identity Switching](#)

16.2.5 Overview of Policy Access Problems After an Application Invokes a Web Service

You may need information to help you diagnose and resolve problems that cause policy access errors that occur after an application invokes a web service.

Topics:

- [Understanding Common Policy Access Problems](#)
- [Solving Common Policy Access Problems](#)

16.2.5.1 Understanding Common Policy Access Problems

If a policy access problem occurs after an application attempts to invoke a web service, then an error message similar to the following is displayed:

- 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.

Policy access issues that occur after an application invokes a web service are commonly caused by the following problems:

- The Policy Manager is down
- The policy URI is missing or the policy name is misspelled.
- The policy does not exist in the repository
- The policy attachment is not in effect due to a cache delay.

16.2.5.2 Solving Common Policy Access Problems

You can use the following information to fix the most common policy access issues:

1. Verify that the Policy Manager is running as described in "[Diagnosing Policy Manager Problems Using the OWSM Policy Manager Page](#)" and [Overview of Common Policy Manager Connection Problems](#)
2. Verify that the mds-owsm datasource connection is reachable and available. For more information, see "Understanding and Managing Data Sources" in *Administering Oracle Fusion Middleware*.
3. Verify that the policy exists in the OWSM 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 Policy Manager Problems Using the OWSM Policy Manager Page](#)
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 OWSM 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 `upgradeWSMRepository()` command. For more information, see

"upgradeWSMRepository" in *WLST Command Reference for Infrastructure Components*.

- Reset the contents of the repository using the `resetWSMRepository` command as described in [Rebuilding the OWSM Repository](#)
- For a custom policy:
 - Import it into the repository as described in [Importing Web Service Policies](#) For information on creating a custom policy, see [Creating a New Web Service Policy](#)
- 6. Check if the user is in a role that has the right permission granted. To modify any roles or permissions, refer to [Modifying the User's Group or Role](#)
- 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 OWSM 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:

- **Initial Cache Refresh**, default 600000 milliseconds (10 minutes)
- **Cache Refresh Time**, default 600000 milliseconds (10 minutes)

For details about tuning these properties, see the following sections:

- [High Availability Configuration and Cache Management Using Fusion Middleware Control](#)
- [Configuring High Availability and Cache Management Using WLST](#)

16.2.6 Overview of Problems Accessing Users in the Credential Store

You may need information to help you diagnose and resolve problems that prevent OWSM from locating a user in the credential store.

Topics:

- [Understanding Common User Access Problems](#)
- [Solving User Access Problems](#)

16.2.6.1 Understanding Common User Access Problems

If OWSM cannot access a user in the credential store, then an error such as the following is displayed:

```
WSM-00015: The user name is missing
```

User access issues are commonly caused by the following problems:

- The credential map `oracle.wsm.security` does not exist in the credential store.
- The user is not listed in the map used by OWSM.
- The CSF key for the entry does not exist in the credential store.

16.2.6.2 Solving User Access Problems

You can use the following information to fix common user access problems.

Verify that the credential map `oracle.wsm.security` exists in the credential store. OWSM 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 [Adding Keys and User Credentials to Configure the Credential Store](#)

If your application uses a credential map other than `oracle.wsm.security`, ensure that any users that OWSM needs to access are duplicated in the `oracle.wsm.security` credential map.

16.2.7 Overview of Common User Authorization Problems After an Application Invokes a Web Service

You may need information to help you diagnose and resolve problems that cause user authorization issues that occur after an application invokes a web service.

Topics:

- [Understanding Common User Authorization Problems](#)
- [Solving a User Authorization Problem](#)

16.2.7.1 Understanding Common User Authorization Problems

If your system fails to authorize a user, then a message similar to the following might be displayed:

```
java.security.AccessControlException: access denied
(oracle.wsm.security.WSFunctionPermission
```

Generally, failure to authorize a user is not really a problem but rather intended behavior; that is, the system was unable to authorize the user for the action that the user was attempting.

16.2.7.2 Solving a User Authorization Problem

You can use the following steps to debug this user authorization issue:

1. 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#updatePartner invoke)
at
java.security.AccessControlContext.checkPermission(AccessControlContext.java:323)
at
java.security.AccessController.checkPermission(AccessController.java:546)
at
oracle.jbo.server.svc.ServicePermissionCheckInterceptor.checkPermission(ServicePermis-
sionCheckInterceptor.java:103)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

2. Pay careful attention to the following information in the log, which is shown in **bold text** in the preceding example. 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 *Securing Applications with Oracle Platform Security Services*.
 - The application stripe name, which is `SalesApp#V2.0` in the preceding log.
 - The permission grant, which is comprised of the Resource name (`http://xmlns.oracle.com/apps/partnerMgmt/partnerCenter/PartnerService#updatePartner` in the log) and Action (`invoke` in the log).

Both of these pieces of information must be specified correctly in your permission grant. For more information, see [Determining Authorization Permissions](#)
3. If your application uses an LDAP-based authenticator and stores all roles in the LDAP, ensure that OWSM can access the users and roles as described in [About Modifying the Default User](#)

16.2.8 Overview of Timestamp Errors After an Application Invokes a Web Service

You may need information to help you diagnose and resolve problems that can cause a timestamp error after an application invokes a web service.

Topics:

- [Understanding the Causes a Timestamp or clockSkew Error](#)
- [Solving Timestamp or clockSkew Errors](#)

16.2.8.1 Understanding the Causes a Timestamp or clockSkew Error

If a timestamp problem occurs after an application invokes a web service, then an error similar to the following is displayed:

```
WSM-00060: Error validating timestamp
```

The problem manifests itself as a timestamp validation or clockSkew error, as follows:

```
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.

16.2.8.2 Solving Timestamp or clockSkew Errors

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 **Clock Skew** as described in [Configuring Security Policy Enforcement Using Fusion Middleware Control](#)
- Set the system clock.
- Use an ntp server to maintain the time.

16.2.9 Overview of Multiple Authentication Security Policy Errors After an Application Invokes a Web Service

You can diagnose and resolve multiple authentication policy errors that occur after an application invokes a web service.

Topics:

- [Understanding Common Multiple Authentication Security Policy Errors](#)
- [Solving Multiple Authentication Security Policy Errors](#)

16.2.9.1 Understanding Common Multiple Authentication Security Policy Errors

If a multiple authentication security policy problem occurs after an application invokes a web service, a multiple policy error (WSM-01823) is displayed in the log. For example, this error appears if multiple authentication policies are attached to a subject.

Multiple authentication policy errors commonly occur when more than one authentication policy is attached to a subject. This situation 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 OWSM 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:/base_domain/serverConfig> displayWSMPolicySet('default-domain-ws-domain_gpa')

Policy Set Details:
-----
Display Name:                default-domain-ws-domain_gpa
Type of Resources:          SOAP Web Service
Scope of Resources:         DOMAIN("base_domain")
Description:                 Global policy attachments for Web Service Endpoint resources.
Enabled:                    true
Policy Reference:           URI=oracle/
wss11_saml_or_username_token_with_message_protection_service_policy, category=security,
enabled=true

wls:/base_domain/serverConfig> displayWSMPolicySet('default-domain-ws-domain')

Policy Set Details:
-----
Display Name:                default-domain-ws-domain
Type of Resources:          SOAP Web Service
Scope of Resources:         DOMAIN("*")
Description:                 Global policy attachments for Web Service Endpoint resources.
Enabled:                    true
Policy Reference:           URI=oracle/wss_saml_or_username_token_service_policy,
category=security, 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.

16.2.9.2 Solving Multiple Authentication Security Policy Errors

Use the following steps to verify if you have multiple policy sets attempting to attach authentication policies:

1. Use the `listWSMPolicySets()` command to display a list of the policy sets in the domain. For more information about this command, see [Viewing a List of Policy Sets](#)
2. Use the `listWSMPolicySubjects` 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 `listWSMPolicySubjects(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:/base_domain/serverConfig> listWSMPolicySubjects(detail=true)

Application: /WLS/base_domain/jaxwsejb30ws

    Assembly: #jaxwsejb

        Subject: WS-Service({http://www.oracle.com/jaxws/tests/
concrete}WsdConcreteService#WsdConcretePort)

            URI="oracle/mex_request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
            URI="oracle/mtom_encode_fault_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
                Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
            URI="oracle/max_request_size_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
                Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
                Property name="max.request.size", value="-1"
            URI="oracle/request_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
                Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
            URI="oracle/soap_request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
            URI="oracle/ws_logging_level_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
```

```

        Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
        Property name="logging.level", value=""
        URI="oracle/test_page_processing_service_policy", category=wsconfig, policy-
status=enabled; source=local policy set; reference-status=enabled; effective=true
        Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
        URI="oracle/wsd1_request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
        Property name="local.policy.reference.source",
value="IMPLIED_FEATURE"
        URI="oracle/
wss11_saml_or_username_token_with_message_protection_service_policy",
category=security, policy-status=enabled; source=global policy set "default-domain-
ws-domain_gpa", scope="DOMAIN('base_domain')"; reference-status=enabled;
effective=true
        URI="oracle/wss_saml_or_username_token_service_policy",
        category=security, policy-status=enabled; source=global policy set "default-domain-
ws-domain", scope="DOMAIN('*')";
reference-status=enabled; effective=true

```

```

        The policy subject is invalid in this context because of the following error:
        WSM-01827 : Attaching multiple policies of category "security/
authentication" is not supported.
Policies in list [ "oracle/
wss11_saml_or_username_token_with_message_protection_service_policy" ]
        already have the category, current policy "oracle/
wss_saml_or_username_token_service_policy" also has that category.

```

3. If you find a conflict, do one of the following:
 - Delete one of the policy sets as described in [Deleting Policy Sets Using WLST](#)
 - Disable one of the policy sets as described in [Enabling and Disabling a Policy Set](#)

For more information, see [Determining the Secure Status of an Endpoint](#)

16.3 Overview of Policy Attachment Issues Using WLST

You use WLST commands to diagnose policy attachment issues.

This section contains the following topics:

- [Understanding the Use of listWSMPolicySubjects Command to Identify Policy Attachment Issues](#)
- [Viewing a Sample Configuration Output with Globally and Directly Attached Policies](#)
- [Viewing a Sample Valid Configuration Output with Directly Attached Policies Only](#)

16.3.1 Understanding the Use of listWSMPolicySubjects Command to Identify Policy Attachment Issues

To ensure that there are no conflicts between directly-attached policies and policies attached globally using policy sets, use the `listWSMPolicySubjects (detail="true")` WLST command.

This command displays a list of the web services or 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 the Web Services in a Domain Using WLST" in *Administering Web Services*.

 **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:

- `listWSMPolicySets()` — Displays a list of the policy sets in the repository. For information about using this command, see [Viewing a List of Policy Sets](#)
- `displayWSMPolicySet()` — Displays the configuration of a specific policy set. For information about using this command, see [Displaying the Configuration of a Policy Set](#)

To view the effective policies for an endpoint using Fusion Middleware Control, see [Viewing Policies Attached to a Web Service Using Fusion Middleware Control](#)

For more information about determining if the endpoint is secure and has a valid configuration, see [Determining the Secure Status of an Endpoint](#)

16.3.2 Viewing a Sample Configuration Output with Globally and Directly Attached Policies

You use the `listWSMPolicySubjects(detail=true)` command for viewing 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.

The global and directly attached security policies are shown in bold.

 **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.

```
Application: /WLS/base_domain/jaxwsejb30ws
```

```
Assembly: #jaxwsejb
```

```
Subject: WS-Service({http://www.oracle.com/jaxws/tests/
concrete}WsdConcreteService#WsdConcretePort)
```

```

URI="oracle/wss_saml_or_username_token_service_policy", category=security,
policy-status=enabled;
source=global policy set "default-domain-ws-domain", scope="DOMAIN('*')";
reference-status=enabled; effective=true
    Property name="reference.priority", value="10"
    URI="oracle/mex_request_processing_service_policy", category=wsconfig,
```

```

policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
  URI="oracle/mtom_encode_fault_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
  URI="oracle/max_request_size_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
  Property name="max.request.size", value="-1"
  URI="oracle/request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
  URI="oracle/soap_request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
  URI="oracle/ws_logging_level_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
  Property name="logging.level", value=""
  URI="oracle/test_page_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
  URI="oracle/wsdl_request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
  URI="oracle/wss10_saml20_token_with_message_protection_service_policy",
category=security, policy-status=enabled; source=local policy set; reference-
status=enabled; effective=false
  Property name="local.policy.reference.source", value="LOCAL_ATTACHMENT"

```

The policy subject is secure in this context.

16.3.3 Viewing a Sample Valid Configuration Output with Directly Attached Policies Only

You use `listWSMPolicySubjects(detail=true)` command for viewing a valid configuration.

The following example shows sample output from the command with directly attached policy shown in bold.

```
Application: /WLS/base_domain/jaxwsejb30ws
```

```
Assembly: #jaxwsejb
```

```
Subject: WS-Service({http://www.oracle.com/jaxws/tests/
concrete}WsdConcreteService#WsdConcretePort)
```

```

  URI="oracle/mex_request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
  URI="oracle/mtom_encode_fault_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
  URI="oracle/max_request_size_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
  Property name="max.request.size", value="-1"
  URI="oracle/request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
  Property name="local.policy.reference.source", value="IMPLIED_FEATURE"

```

```

    URI="oracle/soap_request_processing_service_policy", category=wsconfig,
    policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
    URI="oracle/ws_logging_level_policy", category=wsconfig,
    policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
    Property name="logging.level", value=""
    URI="oracle/test_page_processing_service_policy", category=wsconfig,
    policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
    URI="oracle/wsdl_request_processing_service_policy", category=wsconfig,
    policy-status=enabled; source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="IMPLIED_FEATURE"
    URI="oracle/wss10_saml20_token_with_message_protection_service_policy",
category=security, policy-status=enabled;
source=local policy set; reference-status=enabled; effective=true
    Property name="local.policy.reference.source", value="ANNOTATION"

```

The policy subject is secure in this context.

16.4 About Diagnosing Problems With a Domain Configuration Using WLST

You use WLST commands to diagnose domain configuration issues.

This section contains the following topics:

- [Understanding the Use of checkWSMStatus Command to Identify Domain Configuration Issues](#)
- [Viewing checkWSMStatus Output Showing Status](#)
- [Viewing checkWSMStatus Output Showing Credential Store Failure](#)
- [Viewing checkWSMStatus Output With OAuth2 Global Policy Set Configured](#)
- [Viewing checkWSMStatus Output With OAuth2 Global Policy Set Not Configured](#)

16.4.1 Understanding the Use of checkWSMStatus Command to Identify Domain Configuration Issues

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`), the credential store and keystore configuration (`credstore`), OAuth2 configuration (`oauth2`), and Policy Manager history (`pmHistory`). 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 to function correctly.

For more information about the arguments for this command, see "checkWSMStatus command" in the *WLST Command Reference for Infrastructure Components*.

16.4.2 Viewing checkWSMStatus Output Showing Status

The `checkWSMStatus` command returns the status of the credential store and keys, the Policy Manager, and the enforcement agent for the domain.

The following example illustrates this for the domain `base_domain`:

```
wls:/base_domain/serverConfig> checkWSMStatus (verbose='true')
```

```
Health check for server "EXAMPLESERVER":
```

```
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 "host.example.com:1234".
```

```
Enforcement Agent:
```

```
PASSED.
```

```
Message(s):
```

```
Enforcement is successful.  
Service URL: http://host:port/Diagnostic/DiagnosticService?wsdl
```

```
Health check status on server EXAMPLESERVER is PASSED.
```

```
Health check status for system is PASSED.
```

16.4.3 Viewing checkWSMStatus Output Showing Credential Store Failure

You can use the `checkWSMStatus` command to see the credential store failures.

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',target='EXAMPLESERVER')  
  
Health check for server "EXAMPLESERVER":  
  
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 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 not configured.  
    oracle.wsm.security.SecurityException: WSM-00111 : Keystore is not properly  
configured. Check your keystore configurations.  
Credential Store Diagnostic Messages:  
  Message(s):  
    The alias orakey is either not present in the keystore or is configured  
incorrectly. Check the contents of the keystore and the password for the alias "orakey".  
The password of the alias "orakey" should be the same as the password stored in the csf  
key=sign-csf-key
```

NOTE:- All the above commands are based on the Domain level configurations. The actual alias may have been overridden at runtime due to configuration override.

```
Health check status on server EXAMPLESERVER is FAILED.
```

```
Health check status for system is FAILED.
```

16.4.4 Viewing checkWSMStatus Output With OAuth2 Global Policy Set Configured

The OAuth2 global policy set is Configured for ws-client (SOAP client) subject type. Since the command checks for the OAuth2 related configuration in the GPA attached at the domain level, the steps to create GPA for is also listed.

Example:

```
beginWSMSession();
createWSMPolicySet('oauthTestPolicySet','ws-client','Domain("jrfServer_domain")');
attachWSMPolicy('oracle/http_oauth2_token_client_policy');
attachWSMPolicy('oracle/oauth2_config_client_policy');
setWSMPolicyOverride('oracle/oauth2_config_client_policy','token.uri','http://
myhost.us.example.com:14100/ms_oauth/oauth2/endpoints/oauthservice/tokens');
setWSMPolicyOverride('oracle/
http_oauth2_token_client_policy','oauth2.client.csf.key','basic.client.credentials');
validateWSMPolicySet();
commitWSMSession()

wls:/test_domain1/
serverConfig>checkWSMStatus('oauth2')
```

OAuth2 Client Configuration Status:

```
Message(s):
    OAuth2 Client Configuration Checks for type SOAP Client: PASSED
    Successful OAuth Configurations for Client Type(s): WS_CLIENT
        Health check status on server jrfServer_admin is PASSED.
        Health check status for system is PASSED.
```

16.4.5 Viewing checkWSMStatus Output With OAuth2 Global Policy Set Not Configured

You can view the checkWSMStatus output With OAuth2 global policy set is not configured.

In the following example, no OAuth2 global policy sets are configured.

```
wls:/test_domain1/serverConfig>checkWSMStatus('oauth2')
```

OAuth2 Client Configuration Status:

```
Message(s):
    No OAuth2 client policy (oauth2_config_client_policy or oauth
token policy) attached in the domain for client type(s): REST_CLIENT, WS_CLIENT,
SCA_REST_REFERENCE, SCA_REFERENCE
    Health check for server "jrfServer_admin":
    Health check status on server jrfServer_admin is FAILED.
    Health check status for system is FAILED.
```


16.5 Common Oracle Web Services Manager Exceptions for WS-Trust Use Cases

You can fix the exception and error that occur during an end-to-end WS-Trust use case scenario, study the probable causes and recommended solutions, and use this to diagnose and solve common OWSM exceptions for WS-Trust use cases.

Table 16-1 lists the common OWSM 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 chapters in *Use Cases for Securing Web Services Using Oracle Web Services Manager*:

- Configuring Federation with Microsoft ADFS 2.0 STS as the IP-STS and Oracle STS as the RP-STS
- Configuring Federation with Oracle STS as the IP-STS and Microsoft ADFS 2.0 STS as the RP-STS
- Configuring SAML HOK Using WS-Trust with OpenSSO STS
- Configuring SAML Sender Vouches Using WS-Trust with OpenSSO STS
- Configuring SAML Bearer Using WS-Trust with OpenSSO STS

Table 16-1 Common OWSM Exceptions and Errors for WS-Trust Use Cases

Exception/Error	Possible Cause	Solution
WSM-00015: The user name is missing.	<ol style="list-style-type: none"> 1. The <code>sts.auth.user.csf.key</code> 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 <code>sts.auth.user.csf.key</code> property in the STS issued token client policy with the correct value from the credential store.
<pre>javax.net.ssl.SSLHandshakeException: PKIX path validation failed: java.security.cert.CertPathValidatorException: signature check failed</pre>	<p>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:</p> <pre>JAVA_HOME/jre/lib/security/cacerts</pre> <p>This exception can occur due to either of the following:</p> <ul style="list-style-type: none"> • 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. 	<p>Ensure that a valid SSL certificate for the service has been imported into the trusted keystore at <code>JAVA_HOME/jre/lib/security/cacerts</code>. For more information, see About Configuring Keystores for SSL</p>

Table 16-1 (Cont.) Common OWSM Exceptions and Errors for WS-Trust Use Cases

Exception/Error	Possible Cause	Solution
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 About WS-Trust Configuration
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 Overview of Configuring Keystores for Message Protection
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: <ul style="list-style-type: none"> • SAML Trusted Issuers and DN Lists Using Fusion Middleware Control • Configuring SAML and JWT Trusted Issuers, DN Lists, and Token Attribute Rules Using WLST
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 OWSM. Any STS endpoint that the client is trying to communicate with is protected with a security policy. Oracle STS uses OWSM which provides compatible client and service policies. In this case, the client should not have any trouble finding the corresponding client policy.	The OWSM 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

16.6 Managing third party server integration with OWSM

The Access tokens required to access third party servers become invalid in certain scenarios.

Access tokens do not generally expire, however your access token will become invalid if a user explicitly rejects your application from their settings or if a Twitter admin suspends your application. If your application is suspended, then there will be a note on your application page saying that it has been suspended.



Note:

You should plan for an user's access token invalidity at any time and you will need to re-authorize the user in any such cases.

If and when the token expires, the applications have no way of being notified and will have to rely on the error returned by their applications when invoking Twitter API. In such case when access token has expired, the error code will indicate that and the users will have to use the console (apps.twitter.com) to regenerate access token and update credential store with new value.

If the consumer secret is changed, there is no additional work required. The applications just need to use new consumer secret. Here is an excerpt from Twitter documentation :

If your consumer key and secret become compromised for any reason, you may need to reset it to re-gain secure control of your application's identity and the actions taken in its name. Resetting your consumer key does **not** reset the strings representing your users (their "access tokens") but does make your access tokens invalid when used with the former key. When your existent access tokens are used with your new consumer key and secret, they will continue functioning as expected. Your new consumer key and secret will be represented by completely different strings than its previous incarnation.

Part V

Oracle Web Services Manager Predefined Policies and Assertions Templates

Part V provides reference information describing the predefined policy and assertion templates provided with Oracle Web Services Manager (OWSM).

- [Oracle Web Services Manager Predefined Policies](#), provides an alphabetical listing of all of the predefined policies included with the current release of Oracle Web Services Manager and how to work with them.
- [Oracle Web Services Manager Predefined Assertion Templates](#), describes the predefined assertion templates that you can use to construct your policies or copy to create new policies.

Oracle Web Services Manager Predefined Policies

This chapter describes the Oracle Web Services Manager (OWSM) predefined policies, organized by category. For more information about the predefined policy categories, see "Policy Categories" in *Understanding Oracle Web Services Manager*. For more information about attaching policies, see "[Attaching Policies to Manage and Secure Web Services](#)". This chapter includes the following sections:

Note:

- The predefined policies and assertion templates distributed with the current release are read only. You must copy the policy or assertion template before modifying it; you can copy policies in the security and management categories only. You also have the option of configuring 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](#)".
- When attaching OWSM 14c predefined policies, if you specify a value of blank (" ") in the Value field, the default value will be in effect. If you have imported 11g policies or any custom policies, ensure that the policy has a valid value in the Default field to achieve the same effect; otherwise, the specified value will be picked up.

- [Addressing Policies](#)
- [Atomic Transaction Policies](#)
- [Configuration Policies](#)
- [Management Policies](#)
- [MTOM Policies](#)
- [Reliable Messaging Policies](#)
- [Security Policies-Authentication Only](#)
- [Security Policies-Authorization Only](#)
- [Security Policies-Message Protection Only](#)
- [Security Policies-Message Protection and Authentication](#)
- [Security Policies-Sha256 Only](#)
- [Security Policies—Oracle Entitlements Server](#)
- [SOAP Over JMS Transport Policies](#)

17.1 Addressing Policies

You can use the OWSM predefined addressing policies to check inbound messages for the presence of WS-Addressing headers and effectively disable a globally attached WS Addressing policy at a higher scope.

Topics:

- [oracle/wsaddr_policy](#) checks inbound messages for the presence of WS-Addressing headers conforming to the W3C 2005 Final WS-Addressing Policy standard.
- [oracle/no_addressing_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.

For more information about attaching web services addressing policies, see:

- "Configuring Addressing Using Fusion Middleware Control" in *Administering Web Services*
- "Configuring Addressing Using WLST" in *Administering Web Services*

17.2 Atomic Transaction Policies

You can use the predefined OWSM atomic transaction policies to enable and configure support for atomic transactions.

Topics:

- [oracle/atomic_transaction_policy](#) enables and configures support for atomic transactions.
- [oracle/no_atomic_transaction_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached atomic transaction web service policy at a higher scope.

For more information about attaching web services atomic transaction policies, see:

- "Configuring Atomic Transactions Using Fusion Middleware Control" in *Administering Web Services*
- "Configuring Atomic Transactions Using WLST" in *Administering Web Services*

17.3 Configuration Policies

You can use the OWSM predefined configuration policies to enable and configure web services.

Topics:

 **Note:**

Please note the following:

- Configuration policies cannot be duplicated.
- The assertion templates associated with configuration policies are not available for generating new policies.
- Configuration policies are not supported for SOA composite or Java EE (WebLogic) web services.

- [oracle/async_web_service_policy](#) enables and configures an asynchronous web service.
- [oracle/cache_binary_content_policy](#) enables and configures support for binary caching of content..
- [oracle/fast_infoclient_client_policy](#) enables and configures Fast Infoset on the web service client
- [oracle/fast_infoclient_service_policy](#) enables Fast Infoset on the web service.
- [oracle/max_request_size_policy](#) configures the maximum size, in bytes, of the request message that can be sent to the web service.
- [oracle/mex_request_processing_service_policy](#) enables the exchange of web service metadata.
- [oracle/mtom_encode_fault_service_policy](#) enables the creation of MTOM-enabled SOAP fault messages when MTOM is enabled.
- [oracle/no_async_web_service_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached asynchronous web service policy at a higher scope.
- [oracle/no_cache_binary_content_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached binary caching policy at a higher scope.
- [oracle/no_fast_infoclient_client_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached Fast Infoset client policy at a higher scope.
- [oracle/no_fast_infoclient_service_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached Fast Infoset service policy at a higher scope.
- [oracle/no_max_request_size_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached maximum request size policy at a higher scope.
- [oracle/no_mex_request_processing_service_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached web service metadata exchange policy at a higher scope.
- [oracle/no_mtom_encode_fault_service_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached SOAP fault MTOM encoding policy at a higher scope.
- [oracle/no_persistence_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached persistence policy at a higher scope.

- [oracle/no_pox_http_binding_service_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached Plain Old XML (POX) policy at a higher scope.
- [oracle/no_request_processing_service_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached request processing policy at a higher scope.
- [oracle/no_schema_validation_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached schema validation policy at a higher scope.
- [oracle/no_soap_request_processing_service_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached SOAP request processing policy at a higher scope.
- [oracle/no_test_page_processing_service_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached test page processing policy at a higher scope.
- [oracle/no_ws_logging_level_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached logging policy at a higher scope.
- [oracle/no_wSDL_request_processing_service_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached WSDL request processing policy at a higher scope.
- [oracle/persistence_policy](#) configures the secure conversation persistence mechanism for the web service.
- [oracle/pox_http_binding_service_policy](#) enables an endpoint to receive non-SOAP XML messages that are processed by a user defined `javax.xml.ws.Provider<T>.invoke` method.
- [oracle/request_processing_service_policy](#) enables the web service endpoint to process incoming requests.
- [oracle/schema_validation_policy](#) enables the validation of request messages against the schema.
- [oracle/soap_request_processing_service_policy](#) enables the processing of SOAP requests on the web service endpoint.
- [oracle/test_page_processing_policy](#) enables the Web Service Test Client, as described in "Using the Web Services Test Client" in *Administering Web Services*.
- [oracle/ws_logging_level_policy](#) sets the logging level for diagnostic logs for the web service endpoint.
- [oracle/wSDL_request_processing_service_policy](#) enables access to the WSDL for the web service.

For more information about attaching configuration policies, see:

- "Configuring Web Services Using Fusion Middleware Control" in *Administering Web Services*
- "Configuring Web Services Using WLST" in *Administering Web Services*

17.4 Management Policies

You can use the predefined management policies to log the entire SOAP message for the request and just the SOAP body information for the response.

[oracle/log_policy](#) causes the request, response, and fault messages to be sent to a message log.

17.5 MTOM Policies

You can use the predefined Message Transmission Optimization Mechanism (MTOM) policies to effectively disable a globally attached WS MTOM policy at a higher scope, reject inbound messages that are not in MTOM format, and verifies that outbound messages are in MTOM format..

Topics:

- [oracle/no_mtom_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.
- [oracle/wsmtom_policy](#) rejects inbound messages that are not in MTOM format and verifies that outbound messages are in MTOM format.

For more information about attaching MTOM policies, see:

- "Configuring MTOM Using Fusion Middleware Control" in *Administering Web Services*
- "Configuring MTOM Using WLST" in *Administering Web Services*

17.6 Reliable Messaging Policies

You can use the predefined reliable messaging policies to effectively disables a globally attached Web Services Reliable Messaging policy, configure web services reliable messaging on the web service and client, and configure Web Services Reliable Messaging protocol.

Topics:

- [oracle/no_reliable_messaging_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.
- [oracle/no_wsrm_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.
- [oracle/reliable_messaging_policy](#) configures web services reliable messaging on the web service and client.
- [oracle/wsrm10_policy](#) configures version 1.0 of the Web Services Reliable Messaging protocol.
- [oracle/wsrm11_policy](#) configures version 1.1 of the Web Services Reliable Messaging protocol.

For more information about attaching reliable messaging policies, see:

- "Configuring Reliable Messaging Using Fusion Middleware Control" in *Administering Web Services*
- "Configuring Reliable Messaging Using WLST" in *Administering Web Services*

17.7 Security Policies-Authentication Only

You can use the predefined security policies for authentication only scenarios.

Topics:

 **Note:**

There are no predefined policies for two authentication only scenarios: Kerberos over SSL and SPNEGO. To use these scenarios, create your own policies that use the Kerberos over SSL and SPNEGO assertion templates described in "[Oracle Web Services Manager Predefined Assertion Templates](#)".

- [oracle/wss_saml_bearer_or_username_token_service_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.
- [oracle/wss_saml_or_username_token_service_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.
- [oracle/wss_saml_token_bearer_client_policy](#) includes SAML tokens in outbound SOAP request messages.
- [oracle/http_oam_token_service_policy](#) verifies that the OAM agent has authenticated the user and has established an identity.
- [oracle/http_saml20_token_bearer_client_policy](#) includes a SAML Bearer V2.0 token in the HTTP header.
- [oracle/http_saml20_token_bearer_service_policy](#) authenticates users using credentials provided in the SAML v2.0 token with confirmation method Bearer in the HTTP header.
- [oracle/multi_token_rest_service_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 v2.0 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) token from the HTTP header.
- [oracle/no_authentication_client_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.
- [oracle/no_authentication_service_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.
- [oracle/wss_http_token_client_policy](#) includes credentials in the HTTP header for outbound client requests.

- [oracle/wss_http_token_service_policy](#) uses the credentials in the HTTP header to authenticate users against the OPSS identity store.
- [oracle/wss_username_token_client_policy](#) includes credentials in the WS-Security UsernameToken header for all outbound SOAP request messages.
- [oracle/wss_username_token_service_policy](#) uses the credentials in the UsernameToken WS-Security SOAP header to authenticate users.
- [oracle/wss10_saml_token_client_policy](#) includes SAML tokens in outbound SOAP request messages.
- [oracle/wss10_saml_token_service_policy](#) authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header.
- [oracle/wss10_saml20_token_client_policy](#) includes SAML tokens in outbound SOAP request messages.
- [oracle/wss10_saml20_token_service_policy](#) authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header.
- [oracle/wss11_kerberos_token_client_policy](#) includes a Kerberos token in the WS-Security header in accordance with the WS-Security Kerberos Token Profile v1.1 standard.
- [oracle/wss11_kerberos_token_service_policy](#) extracts the Kerberos token from the SOAP header and authenticates the user.
- [oracle/multi_token_rest_access_service_policy](#) allows access to endpoint with anonymous subject when there is no security token in the request. Also, masks 403 response from service if security token is not present in the request.
- [oracle/multi_token_rest_access_over_ssl_service_policy](#) allows access to endpoint over SSL with anonymous subject when there is no security token in the request. Also, masks 403 response from service if security token is not present in request.
- [oracle/http_anonymous_rest_service_policy](#) allows access to endpoint with anonymous subject in context..
- [oracle/http_anonymous_rest_over_ssl_service_policy](#) allows access to endpoint over SSL with anonymous subject in context.
- [oracle/multi_token_sso_over_ssl_rest_service_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.
- [oracle/multi_token_sso_rest_service_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.

17.8 Security Policies-Authorization Only

You can use predefined security policies for authorization only scenarios.

This section summarizes the predefined OWSM authorization only security policies in the following topics:

- [oracle/binding_authorization_denyall_policy](#) provides a simple role-based authorization policy based on the authenticated subject at the SOAP binding level.
- [oracle/binding_authorization_permitall_policy](#) provides a simple role-based authorization for the request based on the authenticated Subject at the SOAP binding level.
- [oracle/binding_permission_authorization_policy](#) provides a permission-based authorization policy based on the authenticated subject.
- [oracle/component_authorization_denyall_policy](#) provides a simple role-based authorization policy based on the authenticated subject.
- [oracle/component_authorization_permitall_policy](#) provides a simple role-based authorization policy based on the authenticated subject.
- [oracle/component_permission_authorization_policy](#) provides a permission-based authorization policy based on the authenticated Subject.
- [oracle/no_authorization_component_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.
- [oracle/no_authorization_service_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. For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".
- [oracle/whitelist_authorization_policy](#) accepts requests 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.

17.9 Security Policies-Message Protection Only

You can use predefined security policies for message protection only scenarios.

Topics:

- [oracle/no_messageprotection_client_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.
- [oracle/no_messageprotection_service_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.
- [oracle/wss10_message_protection_client_policy](#) provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_message_protection_service_policy](#) enforces message protection (integrity and confidentiality) for inbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss11_message_protection_client_policy](#) provides message integrity and confidentiality for outbound SOAP requests in accordance with the WS-Security 1.1 standard.
- [oracle/wss11_message_protection_service_policy](#) enforces message integrity and confidentiality for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

17.10 Security Policies-Messages Protection and Authentication

OWSM has predefined security policies for message protection and authentication.

This section summarizes these policies in the following topics:

- [oracle/http_basic_auth_over_ssl_client_policy](#) includes credentials in the HTTP header for outbound client requests and verifies that the transport protocol is HTTPS.
- [oracle/http_basic_auth_over_ssl_service_policy](#) uses the credentials in the HTTP header to authenticate users against the OPSS identity store and verifies that the transport protocol is HTTPS.
- [oracle/http_saml20_token_bearer_over_ssl_client_policy](#) includes a SAML Bearer v2.0 token in the HTTP header. The SAML token with confirmation method Bearer is created automatically, and verifies that the transport protocol provides SSL message protection.
- [oracle/http_saml20_bearer_token_over_ssl_service_policy](#) authenticates users using credentials provided in the SAML v2.0 token with confirmation method Bearer in the HTTP header, and verifies that the transport protocol provides SSL message protection.
- [oracle/multi_token_over_ssl_rest_service_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.
 - SPNEGO over HTTP security (non-SSL)-Extracts SPNEGO token information from the HTTP header.
- [oracle/pii_security_policy](#) encrypts the PII data you want to protect.

- [oracle/sts_trust_config_client_policy](#) specifies the STS client configuration information that is used to invoke the STS for token exchange.
- [oracle/sts_trust_config_service_policy](#) specifies the STS configuration information that is used to invoke the STS for token exchange.
- [oracle/wss_saml_or_username_token_over_ssl_service_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.
- [oracle/wss_saml_token_bearer_over_ssl_client_policy](#) includes SAML tokens in outbound SOAP request messages.
- [oracle/wss_saml_token_bearer_over_ssl_service_policy](#) authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.
- [oracle/wss_http_token_over_ssl_client_policy](#) includes credentials in the HTTP header for outbound client requests, authenticates users against the OPSS identity store, and verifies that the transport protocol is HTTPS.
- [oracle/wss_http_token_over_ssl_service_policy](#) extracts the credentials in the HTTP header and authenticates users against the OPSS identity store, and verifies that the transport protocol is HTTPS.
- [oracle/wss_saml_token_over_ssl_client_policy](#) includes SAML tokens in outbound WS-Security SOAP headers using the sender-vouches confirmation type.
- [oracle/wss_saml_token_over_ssl_service_policy](#) enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type, and verifies that the transport protocol provides SSL message protection.
- [oracle/wss_saml20_token_bearer_over_ssl_client_policy](#) includes SAML tokens in outbound SOAP request messages, and verifies that the transport protocol provides SSL message protection.
- [oracle/wss_saml20_token_bearer_over_ssl_service_policy](#) authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header, and verifies that the transport protocol provides SSL message protection.
- [oracle/wss_saml20_token_over_ssl_client_policy](#) includes SAML tokens in outbound WS-Security SOAP headers using the sender-vouches confirmation type, and verifies that the transport protocol provides SSL message protection.
- [oracle/wss_saml20_token_over_ssl_service_policy](#) enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type, and verifies that the transport protocol provides SSL message protection.
- [oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy](#) inserts a SAML bearer assertion issued by a trusted STS.
- [oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy](#) authenticates a SAML bearer assertion issued by a trusted STS.
- [oracle/wss_username_token_over_ssl_client_policy](#) includes credentials in the WS-Security UsernameToken header in outbound SOAP request messages, and verifies that the transport protocol provides SSL message protection.

- [oracle/wss_username_token_over_ssl_service_policy](#) uses the credentials in the WS-Security UsernameToken SOAP header to authenticate users against the OPSS configured identity store, and verifies that the transport protocol provides SSL message protection.
- [oracle/wss_username_token_over_ssl_wssc_client_policy](#) includes credentials in the WS-Security UsernameToken header in outbound SOAP request messages, and verifies that the transport protocol provides SSL message protection.
- [oracle/wss_username_token_over_ssl_wssc_service_policy](#) uses the credentials in the WS-Security UsernameToken SOAP header to authenticate users against the OPSS configured identity store, and verifies that the transport protocol provides SSL message protection.
- [oracle/wss10_saml_hok_token_with_message_protection_client_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.
- [oracle/wss10_saml_hok_token_with_message_protection_service_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.
- [oracle/wss10_saml_token_with_message_integrity_client_policy](#) provides message-level integrity and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_saml_token_with_message_integrity_service_policy](#) enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_saml_token_with_message_protection_client_policy](#) provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_saml_token_with_message_protection_service_policy](#) enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy](#) provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy](#) enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_saml20_token_with_message_protection_client_policy](#) provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_saml20_token_with_message_protection_service_policy](#) enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_username_id_propagation_with_msg_protection_client_policy](#) provides message protection (integrity and confidentiality) and identity propagation for outbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_username_id_propagation_with_msg_protection_service_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.

- [oracle/wss10_username_token_with_message_protection_client_policy](#) provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_username_token_with_message_protection_service_policy](#) enforces message protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy](#) provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy](#) enforces message protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_x509_token_with_message_protection_client_policy](#) provides message protection (integrity and confidentiality) and certificate credential population for outbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss10_x509_token_with_message_protection_service_policy](#) enforces message protection (integrity and confidentiality) and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.
- [oracle/wss11_kerberos_token_with_message_protection_client_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.
- [oracle/wss11_kerberos_token_with_message_protection_service_policy](#) enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard.
- [oracle/wss11_kerberos_token_with_message_protection_basic128_client_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.
- [oracle/wss11_kerberos_token_with_message_protection_basic128_service_policy](#) enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard.
- [oracle/wss11_saml_or_username_token_with_message_protection_service_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.

- [oracle/wss11_saml_token_identity_switch_with_message_protection_client_policy](#) enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.
- [oracle/wss11_saml_token_with_message_protection_client_policy](#) enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.
- [oracle/wss11_saml_token_with_message_protection_wssc_client_policy](#) enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.
- [oracle/wss11_saml_token_with_message_protection_wssc_reauthn_client_policy](#) enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.
- [oracle/wss11_saml_token_with_message_protection_wssc_reauthn_service_policy](#) enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.
- [oracle/wss11_saml20_token_with_message_protection_client_policy](#) enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.
- [oracle/wss11_saml20_token_with_message_protection_service_policy](#) enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.
- [oracle/wss11_sts_issued_saml_hok_with_message_protection_client_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.
- [oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy](#) authenticates a SAML HOK assertion issued by a trusted STS (Security Token Service).
- [oracle/wss11_username_token_with_message_protection_service_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.
- [oracle/wss11_username_token_with_message_protection_client_policy](#) enforces message protection (integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.
- [oracle/wss11_username_token_derivedkey_with_message_protection_service_policy](#) enables use of OWSM to integrate with any backend service client where request contains <wsse11:Salt> or <wsse11:Iteration> element in the username token. These elements are used in Username token to facilitate password-derived keys support. Either signature or encryption is used.
- [oracle/wss11_username_token_derivedkey_with_message_protection_signature_only_client_policy](#) enables use of OWSM to integrate with any backend service which requires <wsse11:Salt> or <wsse11:Iteration> element in the username token. These elements are used in Username token to facilitate password-derived keys support. This client policy is for message protection using signature.
- [oracle/wss11_username_token_derivedkey_with_message_protection_encryption_only_client_policy](#) enables use of OWSM to integrate with any backend service which requires <wsse11:Salt> or <wsse11:Iteration> element in the username token. These elements are used in Username token to facilitate password-derived keys support. This client policy is for message protection using encryption.

- [oracle/wss11_username_token_with_message_protection_wssc_client_policy](#) provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard.
- [oracle/wss11_username_token_with_message_protection_wssc_service_policy](#) enforces message protection (integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.
- [oracle/wss11_x509_token_with_message_protection_client_policy](#) provides message protection (integrity and confidentiality) and certificate-based authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard.
- [oracle/wss11_x509_token_with_message_protection_service_policy](#) enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.
- [oracle/wss11_x509_token_with_message_protection_wssc_client_policy](#) provides message protection (integrity and confidentiality) and certificate-based authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard.
- [oracle/wss11_x509_token_with_message_protection_wssc_service_policy](#) enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

17.11 Security Policies-Sha256 Only

OWSM has predefined sha256 policies.

This section summarizes the predefined OWSM Sha256 only security policies in the following topics:

- [oracle/wss11_saml_or_username_token_with_message_protection_sha256_service_policy](#) enforces message protection (integrity and confidentiality) and an authentication policy, based on whether the client uses a SAML, username, or HTTP token.
- [oracle/wss11_saml_token_identity_switch_with_message_protection_sha256_client_policy](#) enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.
- [oracle/wss11_saml_token_with_message_protection_sha256_client_policy](#) enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.
- [oracle/wss11_saml_token_with_message_protection_sha256_service_policy](#) enables message protection (integrity and confidentiality) and SAML token population for inbound SOAP requests using mechanisms described in WS-Security 1.1.
- [oracle/wss11_username_token_with_message_protection_sha256_client_policy](#) provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard.
- [oracle/wss11_username_token_with_message_protection_sha256_service_policy](#) enforces message protection (integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.
- [oracle/wss_saml_bearer_or_username_token_sha256_service_policy](#) enforces one authentication policy, based on whether the client uses a SAML bearer or username token.
- [oracle/wss_saml_token_bearer_identity_switch_sha256_client_policy](#) performs dynamic identity switching by propagating a different identity than the one based on the authenticated subject.

- [oracle/wss_saml_token_bearer_over_ssl_sha256_client_policy](#) includes SAML tokens in outbound SOAP request messages.
- [oracle/wss_saml_token_bearer_over_ssl_sha256_service_policy](#) authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.
- [oracle/wss_saml_token_bearer_sha256_client_policy](#) includes SAML Bearer tokens in outbound SOAP request messages.
- [oracle/wss_saml_token_bearer_sha256_service_policy](#) authenticates users using credentials provided in SAML Bearer token in the WS-Security SOAP header.

17.12 Security Policies—Oracle Entitlements Server

OWSM has predefined security policies for Oracle Entitlements Server (OES).

Topics:

- [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.

17.13 SOAP Over JMS Transport Policies

You can use predefined policies for SOAP Over JMS Transport.

Topics:

- [oracle/jms_transport_client_policy](#) enables and configures support for SOAP over JMS transport for web service clients.
- [oracle/jms_transport_service_policy](#) enables and configures support for SOAP over JMS transport for web services.
- [oracle/no_jms_transport_client_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached SOAP over JMS transport client policy at a higher scope.
- [oracle/no_jms_transport_service_policy](#) when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached SOAP over JMS transport service policy at a higher scope.

For more information about attaching SOAP over JMS transport policies, see:

- "Configuring SOAP Over JMS Transport Using Fusion Middleware Control" in *Administering Web Services*
- "Configuring SOAP Over JMS Transport Using WLST" in *Administering Web Services*

17.14 oracle/wsaddr_policy

The oracle/wsaddr_policy checks 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.

Display Name: WS Addressing Policy

Category: WS-Addressing

Description

For more information about configuring WS-Addressing on the web service client, see *Web Services Addressing 1.0 - SOAP Binding* specification (<http://www.w3.org/TR/ws-addr-soap/>).

Note:

Please note the following:

- This policy cannot be duplicated.
- The assertion template associated with this policy is not available for generating new policies.
- This policy is not supported for Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-1](#) lists the configuration property that you can override for the addressing policy.

Table 17-1 Configuration Property for oracle/wsaddr_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.15 oracle/no_addressing_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.

Display Name: No Behavior Addressing Policy

Category: WS-Addressing

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

**Note:**

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no_behavior policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-2](#) lists the configuration property that you can override for the no behavior policy.

Table 17-2 Configuration Property for oracle/no_addressing_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.16 oracle/atomic_transaction_policy

The Atomic Transaction Policy enables and configures support for atomic transactions.

Display Name: Atomic Transaction Policy

Category: Atomic Transactions

Description

For more information about atomic transactions, see "Using Web Services Atomic Transactions" in *Developing Oracle Infrastructure Web Services*.

 **Note:**

Please note the following:

- This atomic transactions policy cannot be duplicated.
- The assertion template associated with this atomic transactions policy is not available for generating new policies.
- This atomic transactions policy is not supported for Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-3](#) lists the configuration properties that you can override for atomic transactions.

Table 17-3 Configuration Properties for oracle/atomic_transaction_policy

Name	Description	Default	Required?
flow.type	Whether the web services atomic transaction coordination context is passed with the transaction flow. Valid values include: <ul style="list-style-type: none"> • MANDATORY • NEVER • SUPPORTS For more information about the valid values, see "Configuring Web Service Atomic Transactions" in <i>Developing Oracle Infrastructure Web Services</i> .	SUPPORTS	Optional
version	Version of the web services atomic transaction coordination context that is supported. For web service clients, it specifies the version used for outbound messages only. The value specified must be consistent across the entire transaction. Valid values include: <ul style="list-style-type: none"> • DEFAULT • WSAT10 • WSAT11 • WSAT12 For more information about the valid values, see "Configuring Web Service Atomic Transactions" in <i>Developing Oracle Infrastructure Web Services</i> .	DEFAULT	Optional
reference.priority	See " reference.priority ".	None	Optional

17.17 oracle/no_atomic_transaction_policy

When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached atomic transaction web service policy at a higher scope.

Display Name: No Atomic Transaction Policy

Category: Atomic Transactions

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

For more information about atomic transactions, see "Using Web Services Atomic Transactions" in *Developing Oracle Infrastructure Web Services*.

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no_behavior policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-4](#) lists the configuration property that you can override for the no behavior policy.

Table 17-4 Configuration Property for oracle/no_atomic_transaction_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.18 oracle/async_web_service_policy

The Async Web Service Policy enables and configures an asynchronous web service.

Display Name: Async Web Service Policy

Category: Configuration

Description

Enables and configures an asynchronous web service.

Note:

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-5](#) lists the configuration properties that you can override for asynchronous web services.

Table 17-5 Configuration Property for oracle/async_web_service_policy

Name	Description	Default	Required?
jms.access.user	The user that is authorized to use the JMS queues. 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 Using Fusion Middleware Control" in <i>Administering Web Services</i> .	OracleSystemUser	Optional
jms.connection.factory	Name of the connection factory for the JMS request queue.	weblogic.jms.XAConnectionFactory (default JMS connection factory)	Optional
jms.queue	Name of the request queue.	oracle.j2ee.ws.server.async.DefaultRequestQueue	Optional
jms.response.connection.factory	Name of the connection factory for the JMS response queue.	weblogic.jms.XAConnectionFactory (default JMS connection factory)	Optional
jms.response.queue	Name of the request queue.	oracle.j2ee.ws.server.async.DefaultResponseQueue	Optional

Table 17-5 (Cont.) Configuration Property for oracle/async_web_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.19 oracle/cache_binary_content_policy

The oracle/cache_binary_content_policy enables and configures support for binary caching of content.

Display Name: Cache Binary Content Policy

Category: Configuration

Description

Enables and configures support for binary caching of content.

Note:

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-6](#) lists the configuration properties that you can override for binary caching.

Table 17-6 Configuration Properties for oracle/cache_binary_content_policy

Name	Description	Default	Required?
mode	Value that specifies the runtime requirements of XTI scalable DOM in OraSAAJ. Valid values include: <ul style="list-style-type: none"> com.oracle.webservices.api.CacheBinaryContentMode.BINARY—Fastest, but most memory intensive. Not recommended for production. com.oracle.webservices.api.CacheBinaryContentMode.FILE—One temporary file per document. Recommended approach. Need to specify the directory in which to store the temporary files as arg1. com.oracle.webservices.api.CacheBinaryContentMode.BLOB—Slowest. Need to specify the URL of the DBMS connection as arg1. 	BINARY	Optional
arg1	Boolean value that defines one of the following values: <ul style="list-style-type: none"> If mode is set to BINARY, this argument is not required. If mode is set to FILE, specifies the directory in which to store the temporary files as arg1. If mode is set to BLOB, specifies the URL of the DBMS connection. 	java.io.tmpdir	Optional
reference.priority	See "reference.priority".	None	Optional

17.20 oracle/fast_infoset_client_policy

The oracle/fast_infoset_client_policy enables and configures Fast Infoset on the web service client.

Display Name: Fast Infoset Client Policy

Category: Configuration

Description

Enables and configures Fast Infoset on the web service client.

For more information about Fast Infoset, see:

- JAX-WS Web Services: "Optimizing XML Transmission Using Fast Infoset" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.
- Oracle Infrastructure Web Services: "Optimizing XML Transmission Using Fast Infoset" in *Developing Oracle Infrastructure Web Services*.

 **Note:**

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-7](#) lists the configuration properties that you can override for Fast Infoset clients.

Table 17-7 Configuration Properties for oracle/fastinfoset_client_policy

Name	Description	Default	Required?
fast.infoset.content.negotiation	Value that specifies the Fast Infoset content negotiation setting. Valid values include: <ul style="list-style-type: none"> • OPTIMISTIC—Assumes that Fast Infoset is enabled on the service. • PESSIMISTIC—Initial request from client is sent without Fast Infoset enabled. If it is determined that Fast Infoset is enabled on the service, subsequent requests will be sent with FastInfoset enabled on the client. • NONE—Client does not support Fast Infoset. 	NONE	Optional
reference.priority	See " reference.priority ".	None	Optional

17.21 oracle/fast_infoset_service_policy

The oracle/fast_infoset_service_policy enables Fast Infoset on the web service.

Display Name: Fast Infoset Service Policy

Category: Configuration

Description

Enables Fast Infoset on the web service.

For more information about Fast Infoset, see:

- JAX-WS Web Services: "Optimizing XML Transmission Using Fast Infoset" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.

- Oracle Infrastructure Web Services: "Optimizing XML Transmission Using Fast Infoset" in *Developing Oracle Infrastructure Web Services*.

 **Note:**

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-8](#) lists the configuration properties that you can override for Fast Infoset web services.

Table 17-8 Configuration Properties for oracle/fastinfoset_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.22 oracle/max_request_size_policy

The oracle/max_request_size_policy configures the maximum size, in bytes, of the request message that can be sent to the web service.

Display Name: Max Request Size Policy

Category: Configuration

Description

Configures the maximum size, in bytes, of the request message that can be sent to the web service.

 **Note:**

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-9](#) lists the configuration properties that you can override when enabling maximum request size on the web service.

Table 17-9 Configuration Properties for oracle/max_request_size_policy

Name	Description	Default	Required?
max.request.size	Maximum size of the request message, in bytes. A value of -1 indicates that there is no maximum request size.	-1	Optional
reference.priority	See " reference.priority ".	None	Optional

17.23 oracle/mex_request_processing_service_policy

The oracle/mex_request_processing_service_policy enables the exchange of web service metadata.

Display Name: MEX Request Processing Service Policy

Category: Configuration

Description

Enables the exchange of web service metadata.

 **Note:**

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-10](#) lists the configuration properties that you can override when enabling the exchange of web service metadata.

Table 17-10 Configuration Properties for oracle/mex_request_processing_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.24 oracle/mtom_encode_fault_service_policy

The oracle/mtom_encode_fault_service_policy enables the creation of MTOM-enabled SOAP fault messages when MTOM is enabled.

Display Name: MTOM Encode Fault Service Policy

Category: Configuration

Description

Enables the creation of MTOM-enabled SOAP fault messages when MTOM is enabled.

 **Note:**

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-11](#) lists the configuration properties that you can override when enabling MTOM encoding for SOAP faults.

Table 17-11 Configuration Properties for oracle/mtom_encode_fault_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.25 oracle/no_async_web_service_policy

The oracle/no_async_web_service_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached asynchronous web service policy at a higher scope.

Display Name: No Async Web Service Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-12](#) lists the configuration property that you can override for the no behavior policy.

Table 17-12 Configuration Property for oracle/no_async_web_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.26 oracle/no_cache_binary_content_policy

The oracle/no_cache_binary_content_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached binary caching policy at a higher scope.

Display Name: No Cache Binary Content Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-13](#) lists the configuration property that you can override for the no behavior policy.

Table 17-13 Configuration Property for oracle/no_cache_binary_content_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.27 oracle/no_fast_infoset_client_policy

The oracle/no_fast_infoset_client_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached Fast Infoset client policy at a higher scope.

Display Name: No Fast Infoset Client Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-14](#) lists the configuration property that you can override for the no behavior policy.

Table 17-14 Configuration Property for oracle/no_fast_infoset_client_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.28 oracle/no_fast_infoset_service_policy

The oracle/no_fast_infoset_service_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached Fast Infoset service policy at a higher scope.

Display Name: No Fast Infoset Service Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-15](#) lists the configuration property that you can override for the no behavior policy.

Table 17-15 Configuration Property for oracle/no_fast_infoset_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.29 oracle/no_max_request_size_policy

The oracle/no_max_request_size_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached maximum request size policy at a higher scope.

Display Name: No Max Request Size Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

 **Note:**

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-16](#) lists the configuration property that you can override for the no behavior policy.

Table 17-16 Configuration Property for oracle/no_max_request_size_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.30 oracle/no_mex_request_processing_service_policy

The oracle/no_mex_request_processing_service_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached web service metadata exchange policy at a higher scope.

Display Name: No MEX Request Processing Service Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

 **Note:**

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-17](#) lists the configuration property that you can override for the no behavior policy.

Table 17-17 Configuration Property for oracle/no_mex_request_processing_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.31 oracle/no_mtom_encode_fault_service_policy

The oracle/no_mtom_encode_fault_service_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached SOAP fault MTOM encoding policy at a higher scope.

Display Name: No MTOM Encode Fault Service Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-18](#) lists the configuration property that you can override for the no behavior policy.

Table 17-18 Configuration Property for oracle/no_mtom_encode_fault_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.32 oracle/no_persistence_policy

The oracle/no_persistence_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached persistence policy at a higher scope.

Display Name: No Persistence Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-19](#) lists the configuration property that you can override for the no behavior policy.

Table 17-19 Configuration Property for oracle/no_persistence_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.33 oracle/no_pox_http_binding_service_policy

The oracle/no_pox_http_binding_service_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached Plain Old XML (POX) policy at a higher scope.

Display Name: No Pox Http Binding Service Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-20](#) lists the configuration property that you can override for the no behavior policy.

Table 17-20 Configuration Property for oracle/no_pox_http_binding_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.34 oracle/no_request_processing_service_policy

The oracle/no_request_processing_service_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached request processing policy at a higher scope.

Display Name: No Request Processing Service Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

**Note:**

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-21](#) lists the configuration property that you can override for the no behavior policy.

Table 17-21 Configuration Property for oracle/no_request_processing_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.35 oracle/no_schema_validation_policy

The oracle/no_schema_validation_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached schema validation policy at a higher scope.

Display Name: No Schema Validation Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

 **Note:**

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-22](#) lists the configuration property that you can override for the no behavior policy.

Table 17-22 Configuration Property for oracle/no_schema_validation_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.36 oracle/no_soap_request_processing_service_policy

The oracle/no_soap_request_processing_service_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached SOAP request processing policy at a higher scope.

Display Name: No Soap Request Processing Service Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

 **Note:**

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-23](#) lists the configuration property that you can override for the no behavior policy.

Table 17-23 Configuration Property for oracle/no_soap_request_processing_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.37 oracle/no_test_page_processing_service_policy

The oracle/no_test_page_processing_service_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached test page processing policy at a higher scope.

Display Name: No Test Page Processing Service Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".



Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-24](#) lists the configuration property that you can override for the no behavior policy.

Table 17-24 Configuration Property for oracle/no_test_page_processing_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.38 oracle/no_ws_logging_level_policy

The oracle/no_ws_logging_level_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached logging policy at a higher scope.

Display Name: No Ws Logging Level Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-25](#) lists the configuration property that you can override for the no behavior policy.

Table 17-25 Configuration Property for oracle/no_ws_logging_level_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.39 oracle/no_wsdl_request_processing_service_policy

The oracle/no_wsdl_request_processing_service_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached WSDL request processing policy at a higher scope.

Display Name: No Wsdl Request Processing Service Policy

Category: Configuration

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no behavior policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-26](#) lists the configuration property that you can override for the no behavior policy.

Table 17-26 Configuration Property for oracle/no_wsdl_request_processing_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.40 oracle/persistence_policy

The oracle/persistence_policy configures the secure conversation persistence mechanism for the web service.

Display Name: Persistence Policy

Category: Configuration

Description

Configures the secure conversation persistence mechanism for the web service.

Note:

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-27](#) lists the configuration properties that you can override when enabling the policy.

Table 17-27 Configuration Properties for oracle/persistence_policy

Attribute	Description	Default	Required?
providerName	Identifies the persistence provider registered in the system. Possible values are: <ul style="list-style-type: none"> • oracle:jrf:Memory is the in-memory-based persistence provider. • oracle:jrf:Coherence is the integrated Coherence provider. Note: For J2SE clients, you can configure oracle:jrf:Memory only.	oracle:jrf:Coherence, when available.	Optional
reference.priority	See " reference.priority ".	None	Optional

17.41 oracle/pox_http_binding_service_policy

The oracle/pox_http_binding_service_policy enables an endpoint to receive non-SOAP XML messages that are processed by a user defined.

Display Name: Pox Http Binding Service Policy

Category: Configuration

Description

Enables an endpoint to receive non-SOAP XML messages that are processed by a user defined `javax.xml.ws.Provider<T>.invoke` method.

 **Note:**

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-28](#) lists the configuration property that you can override when enabling the policy.

Table 17-28 Configuration Property for oracle/pox_http_binding_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.42 oracle/request_processing_service_policy

The oracle/request_processing_service_policy enables the web service endpoint to process incoming requests.

Display Name: Request Processing Service Policy

Category: Configuration

Description

Enables the web service endpoint to process incoming requests.

 **Note:**

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-29](#) lists the configuration property that you can override when enabling this policy.

Table 17-29 Configuration Property for oracle/request_processing_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.43 oracle/schema_validation_policy

The oracle/schema_validation_policy enables the validation of request messages against the schema.

Display Name: Schema Validation Policy

Category: Configuration

Description

Enables the validation of request messages against the schema.

**Note:**

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-30](#) lists the configuration property that you can override when enabling this policy.

Table 17-30 Configuration Property for oracle/schema_validation_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.44 oracle/soap_request_processing_service_policy

The oracle/soap_request_processing_service_policy enables the processing of SOAP requests on the web service endpoint.

Display Name: Soap Request Processing Service Policy

Category: Configuration

Description

Enables the processing of SOAP requests on the web service endpoint.

Note:

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-31](#) lists the configuration property that you can override when enabling this policy.

Table 17-31 Configuration Property for oracle/soap_request_processing_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.45 oracle/test_page_processing_policy

The oracle/test_page_processing_policy enables the Web Service Test Client. It contains `reference.priority` as configuration property.

Display Name: Test Page Processing Service Policy

Category: Configuration

Description

Enables the Web Service Test Client, as described in "Using the Web Services Test Client" in *Administering Web Services*.



Note:

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-32](#) lists the configuration property that you can override when enabling this policy.

Table 17-32 Configuration Property for oracle/test_page_processing_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.46 oracle/ws_logging_level_policy

The oracle/ws_logging_level_policy sets the logging level for diagnostic logs for the web service endpoint. It contains logging.level and reference.priority as configuration properties.

Display Name: Ws Logging Level Policy

Category: Configuration

Description

Sets the logging level for diagnostic logs for the web service endpoint.

 **Note:**

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-33](#) lists the configuration properties that you can override when enabling this policy.

Table 17-33 Configuration Property for oracle/ws_logging_level_policy

Name	Description	Default	Required?
logging.level	Defines the logging level. Valid values include: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, or NULL.	None	Optional
reference.priority	See " reference.priority ".	None	Optional

17.47 oracle/wSDL_request_processing_service_policy

The oracle/wSDL_request_processing_service_policy enables access to the WSDL for the web service. It contains `reference.priority` as configuration property.

Display Name: WSDL Request Processing Service

Category: Configuration

Description

Enables access to the WSDL for the web service.

 **Note:**

Please note the following:

- This configuration policy cannot be duplicated.
- The assertion template associated with this configuration policy is not available for generating new policies.
- This configuration policy is not supported for SOA composite or Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-34](#) lists the configuration property that you can override when enabling this policy.

Table 17-34 Configuration Property for oracle/ws_logging_level_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.48 oracle/log_policy

The oracle/log_policy causes the request, response, and fault messages to be sent to a message log. By default, this policy logs the entire SOAP message for the request and just the SOAP body information for the response.

Display Name: Log Policy

Category: Management

Description

Messages are logged to the message log for the domain. For information about viewing and filtering message logs, see "Using Message Logs for Web Services" in *Administering Web Services*.

 **Note:**

This policy is not supported for Java EE (WebLogic) web services.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/security_log_template](#)

The assertion is not advertised in the WSDL.

Configuration

[Table 17-35](#) lists the configuration property that you can override for the log policy.

Table 17-35 Configuration Property for oracle/log_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.49 oracle/no_mtom_policy

The oracle/no_mtom_policy is a no behavior policy. When directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached MTOM policy at a higher scope.

Display Name: No Behavior MTOM Policy

Category: MTOM Attachments

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".



Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no_behavior policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-36](#) lists the configuration property that you can override for the no behavior policy.

Table 17-36 Configuration Property for oracle/no_mtom_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.50 oracle/wsmtom_policy

The oracle/wsmtom_policy rejects inbound messages that are not in MTOM format and verifies that outbound messages are in MTOM format. MTOM defines a method for optimizing the transmission of XML data of type `xs:base64Binary` or `xs:hexBinary` in SOAP messages.

Display Name: WS MTOM Policy

Category: MTOM Attachments

Description

For more information about MTOM, see the following specifications for SOAP 1.2 and 1.1., respectively: <http://www.w3.org/TR/2005/REC-soap12-mtom-20050125> and <http://www.w3.org/Submission/2006/SUBM-soap11mtom10-20060405>.

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 );
    }
}
```

Note:

Please note the following:

- This MTOM policy cannot be duplicated.
- The assertion template associated with this policy is not available for generating new policies.
- This policy is not supported for Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-37](#) lists the configuration property that you can override for the MTOM policy.

Table 17-37 Configuration Property for oracle/wsmtom_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.51 oracle/no_reliable_messaging_policy

The oracle/no_reliable_messaging_policy is a no behavior 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.

Display Name: No Reliable Messaging Policy

Category: Reliable Messaging

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

For more information about reliable messaging, see "Using Web Services Atomic Transactions" in *Developing Oracle Infrastructure Web Services*.



Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no_behavior policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-38](#) lists the configuration property that you can override for the no behavior policy.

Table 17-38 Configuration Property for oracle/no_reliable_messaging_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.52 oracle/no_wsrp_policy

The oracle/no_wsrp_policy is a no behavior 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.

Display Name: No Behavior RM Policy

Category: Reliable Messaging

Note:

This policy has been deprecated. Oracle recommends that you use the `oracle/no_reliable_messaging` policy, as described in "[oracle/no_reliable_messaging_policy](#)".

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no_behavior policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-39](#) lists the configuration property that you can override for the no behavior policy.

Table 17-39 Configuration Property for oracle/no_wsrp_policy

Name	Description	Default	Required?
<code>reference.priority</code>	See " reference.priority ".	None	Optional

17.53 oracle/reliable_messaging_policy

The oracle/reliable_messaging_policy configures web services reliable messaging on the web service and client. This policy can be attached to any SOAP-based web service and client.

Display Name: Reliable Messaging Policy

Category: Reliable Messaging

Description

The web service client will automatically detect the WSDL policy assertions at run time and use them to enable the advertised version of reliable messaging on the client. When more than one version is enabled, the generated WSDL has policy alternatives for the given versions, which enables the client to select any version. The client must consistently use the selected version of the protocol for all interaction with a given sequence.

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 reliable messaging sequence boundaries.

The following example shows sample client code for web services reliable messaging for a servlet client. In this example, 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`).

```
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());
        }
    }
}
```

 **Note:**

Please note the following:

- This reliable messaging policy cannot be duplicated.
- The assertion template associated with this policy is not available for generating new policies.
- This policy is not supported for Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-40](#) lists the configuration properties that you can override when enabling the policy.

Table 17-40 Configuration Properties for oracle/reliable_messaging_policy

Name	Description	Default	Required?
acknowledgement.interval	<p>Maximum interval, in milliseconds, in which the destination endpoint must transmit a standalone acknowledgement.</p> <p>The value specified must be a positive value and conform to the XML schema duration lexical format, <i>PnYnMnDTnHnMnS</i>, where <i>nY</i> specifies the number of years, <i>nM</i> specifies the number of months, <i>nD</i> specifies the number of days, <i>T</i> is the date/time separator, <i>nH</i> specifies the number of hours, <i>nM</i> specifies the number of minutes, and <i>nS</i> specifies the number of seconds.</p> <p>This value is set at sequence creation time, and cannot be reset.</p>	P0DT0.2S (200 milliseconds)	Optional
destination.allowed.versions	<p>Reliable messaging version(s) supported.</p> <p>When more than one version is enabled, the generated WSDL will list policy alternatives for the given versions, allowing the client to select the version. The client must use the selected version consistently for all interactions in a given sequence.</p> <p>Valid values include:</p> <ul style="list-style-type: none"> • DEFAULT (supports all versions) • WS_RM_1_0 • WS_RM_1_1 • WS_RM_1_2 	DEFAULT	Optional

Table 17-40 (Cont.) Configuration Properties for oracle/reliable_messaging_policy

Name	Description	Default	Required?
destination.non.buffered	<p>Flag indicating that non-buffered receipt of messages is requested.</p> <p>This value is set at sequence creation time, and cannot be reset.</p>	false	Optional
inactivity.timeout	<p>Number of milliseconds which defines an inactivity interval. After this amount of time, if the destination endpoint has not received a message from the source endpoint, the destination endpoint may consider the sequence to have terminated due to inactivity. The same is true for the source endpoint. By default, sequences never timeout.</p> <p>Implementations of RM source and RM destination are free to manage resources associated with the sequence as desired, but there are no guarantees that the sequence will be usable by either party after the inactivity timeout expires.</p> <p>The value specified must be a positive value and conform to the XML schema duration lexical format, <i>PnYnMnDTnHnMnS</i>, where <i>nY</i> specifies the number of years, <i>nM</i> specifies the number of months, <i>nD</i> specifies the number of days, <i>T</i> is the date/time separator, <i>nH</i> specifies the number of hours, <i>nM</i> specifies the number of minutes, and <i>nS</i> specifies the number of seconds.</p> <p>Set at sequence creation time, and cannot be reset.</p>	P0DT600S (600 seconds)	Optional
max.retry.count	<p>Number of times that the JMS queue on the invoked WebLogic Server instance attempts to deliver the message to the web service implementation until the operation is successfully invoked.</p>	-1	Optional

Table 17-40 (Cont.) Configuration Properties for oracle/reliable_messaging_policy

Name	Description	Default	Required?
optional	<p>Flag that specifies whether reliable messaging is required.</p> <p>This flag enables a service endpoint to support reliable or non-reliable communication with different clients.</p> <p>If <code>optional</code> is set to <code>false</code>, then every message sent to a service must be reliable. If <code>optional</code> is set to <code>true</code>, then a client can choose to send requests with or without the WS-RM protocol. In this case, the service is required to handle either.</p> <p>When used in combination with an operation-level "required" WS-RM policy, operations without an explicit WS-RM policy do not need to be called with the WS-RM protocol, but operations with an explicit WS-RM policy must be called with the WS-RM protocol.</p>	false	Optional
reference.priority	See " reference.priority ".	None	Optional
sequence.q.o.s	<p>Delivery assurance for reliable messaging.</p> <p>Valid values include:</p> <ul style="list-style-type: none"> EXACTLY_ONCE—Every message is delivered exactly once, without duplication. AT_MOST_ONCE—Messages are delivered at most once, without duplication. It is possible that some messages may not be delivered at all. AT_LEAST_ONCE—Every message is delivered at least once. It is possible that some messages are delivered more than once. UNSPECIFIED 	EXACTLY_ONCE	Optional
sequence.in.order	Flag that specifies that messages are delivered in the order that they were sent.	false	Optional

Table 17-40 (Cont.) Configuration Properties for oracle/reliable_messaging_policy

Name	Description	Default	Required?
sequence.expiration	<p>Amount of time after which the reliable web service expires and does not accept any new sequence messages.</p> <p>If this limit is reached before the sequence naturally completes, it will be forcibly terminated.</p> <p>The value specified must be a positive value and conform to the XML schema duration lexical format, <i>PnYnMnDTnHnMnS</i>, where <i>nY</i> specifies the number of years, <i>nM</i> specifies the number of months, <i>nD</i> specifies the number of days, <i>T</i> is the date/time separator, <i>nH</i> specifies the number of hours, <i>nM</i> specifies the number of minutes, and <i>nS</i> specifies the number of seconds.</p> <p>This value is set at sequence creation time, and cannot be reset.</p>	P1D (1 day)	Optional
sequence.s.t.r	<p>Flag that specifies that in order to secure messages in a reliable sequence, the runtime will use the <code>wsse:SecurityTokenReference</code> that is referenced in the <code>CreateSequence</code> message.</p>	false	Optional
sequence.transport.security	<p>Flag that specifies that in order to secure messages in a reliable sequence, the RM Sequence must be bound to the session(s) of the underlying transport-level protocol used to carry the <code>CreateSequence</code> and <code>CreateSequenceResponse</code> message.</p> <p>When present, this assertion must be used in conjunction with the <code>sp:TransportBinding</code> assertion.</p>	false	Optional

Table 17-40 (Cont.) Configuration Properties for oracle/reliable_messaging_policy

Name	Description	Default	Required?
source.backoff.algorithm	<p>Backoff algorithm.</p> <p>If a destination endpoint does not acknowledge a sequence of messages for the time interval specified by the base retransmission interval (<code>source.base.retransmission.interval</code>), the configured backoff algorithm is used for timing successive retransmissions by the source endpoint, should the message continue to go unacknowledged.</p> <p>Valid values include:</p> <ul style="list-style-type: none"> • EXPONENTIAL—Successive retransmission intervals increase exponentially, based on the base retransmission interval. For example, if the base retransmission interval is 2 seconds, and the exponential backoff element is set, successive retransmission intervals if messages continue to go unacknowledged are 2, 4, 8, 16, 32, and so on. • CONSTANT—Same retransmission interval is used in successive retries. • NONE <p>This value is set at sequence creation time, and cannot be reset.</p>	NONE	Optional
source.base.retransmission.interval	<p>Interval of time that must pass before a message will be retransmitted to the RM destination (in the event a prior transmission failed.)</p> <p>This interval can be used in conjunction with the backoff algorithm (<code>source.backoff.algorithm</code>) to specify the algorithm that is used to adjust the retransmission interval.</p> <p>The value specified must be a positive value and conform to the XML schema duration lexical format, <i>PnYnMnDTnHnMnS</i>, where <i>nY</i> specifies the number of years, <i>nM</i> specifies the number of months, <i>nD</i> specifies the number of days, <i>T</i> is the date/time separator, <i>nH</i> specifies the number of hours, <i>nM</i> specifies the number of minutes, and <i>nS</i> specifies the number of seconds.</p> <p>This value is set at sequence creation time, and cannot be reset.</p>	P0DT3S	Optional

Table 17-40 (Cont.) Configuration Properties for oracle/reliable_messaging_policy

Name	Description	Default	Required?
source.version	<p>Reliable messaging version(s) supported by the RM source.</p> <p>When the service WSDL contains policy alternatives for multiple RM versions, the client can select the version via this attribute. If the WSDL contains multiple RM versions and this attribute is not explicitly set, then either RM 1.2 is used or the highest version in the WSDL, if the WSDL does not contain RM 1.2.</p> <p>Valid values include:</p> <ul style="list-style-type: none"> • DEFAULT (supports all versions) • WS_RM_1_0 • WS_RM_1_1 • WS_RM_1_2 <p>If the WSDL contains only one RM version, this attribute is ignored and the version in the WSDL is used.</p> <p>Other possible values are DEFAULT, WS_RM_1_0, and WS_RM_1_1.</p>	WS_RM_1_2	Optional
reference.priority	See " reference.priority ".	None	Optional

17.54 oracle/wsrml10_policy

The oracle/wsrml10_policy configures version 1.0 of the Web Services Reliable Messaging protocol. This policy can be attached to any SOAP-based client or endpoint.

Display Name: WS RM10 Policy

Category: Reliable Messaging

Note:

This policy has been deprecated. Oracle recommends that you use the `oracle/reliable_messaging` policy, as described in "[oracle/reliable_messaging_policy](#)".

Description

The web service client will automatically detect the WSDL policy assertions at run time and use them to enable the advertised version of reliable messaging on the client.

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 reliable messaging sequence boundaries.

The example in [oracle/wsrmtom_policy](#) illustrates a servlet client. In this example, 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`).

 **Note:**

Please note the following:

- This reliable messaging policy cannot be duplicated.
- The assertion template associated with this policy is not available for generating new policies.
- This policy is not supported for Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-41](#) lists the configuration properties that you can override for the reliable messaging policy.

Table 17-41 Configuration Properties for the wsrml0_policy

Name	Description	Default	Required
DeliveryAssurance	<p>Delivery assurance. The following defines the delivery assurance types:</p> <ul style="list-style-type: none"> 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. <p>In addition, you can configure whether messages are delivered in the order that they were sent.</p> <p>Valid values include</p> <ul style="list-style-type: none"> AtLeastOnce AtLeastOnceInOrder AtMostOnce AtMostOnceInOrder ExactlyOnce ExactlyOnceInOrder InOrder 	InOrder	Optional
StoreType	<p>Type of message store.</p> <p>Valid values include:</p> <ul style="list-style-type: none"> FileSystem (not fully supported) InMemory JDBC 	InMemory	Optional
StoreName	Name of the message store.	oracle	Optional
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	Optional
InactivityTimeout	<p>Number of milliseconds which defines an inactivity interval. After this amount of time, if the destination endpoint has not received a message from the source endpoint, the destination endpoint may consider the sequence to have terminated due to inactivity. The same is true for the source endpoint. By default, sequences never timeout.</p> <p>Implementations of RM source and RM destination are free to manage resources associated with the sequence as desired, but there are no guarantees that the sequence will be usable by either party after the inactivity timeout expires.</p>	600000	Optional

Table 17-41 (Cont.) Configuration Properties for the wsrml1_policy

Name	Description	Default	Required
BaseRetransmissionInterval	Interval of time that must pass before a message will be retransmitted to the RM destination (in the event a prior transmission failed.)	3000	Optional

17.55 oracle/wsrml1_policy

The oracle/wsrml1_policy configures version 1.1 of the Web Services Reliable Messaging protocol. This policy can be attached to any SOAP-based client or endpoint.

Display Name: WS RM11 Policy

Category: Reliable Messaging

Note:

This policy has been deprecated. Oracle recommends that you use the `oracle/reliable_messaging` policy, as described in "[oracle/reliable_messaging_policy](#)".

Description

The web service client will automatically detect the WSDL policy assertions at run time and use them to enable the advertised version of reliable messaging on the client.

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 reliable messaging sequence boundaries.

The example in [oracle/reliable_messaging_policy](#) illustrates a servlet client. In this example, 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`).

Note:

Please note the following:

- This reliable messaging policy cannot be duplicated.
- The assertion template associated with this policy is not available for generating new policies.
- This policy is not supported for Java EE (WebLogic) web services.

Assertion

An assertion template is not provided for creating this policy. For that reason, it is important that you do not delete this policy. To recreate it you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-41](#) lists the configuration properties that you can override for this policy

17.56 oracle/http_basic_auth_over_ssl_client_policy

The oracle/http_basic_auth_over_ssl_client_policy includes credentials in the HTTP header for outbound client requests and verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused.

Display Name: HTTP Basic Auth Over SSL Client Policy

Category: Security

Description

This policy can be enforced on any HTTP-based client endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_http_token_over_ssl_client_template](#)

The assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-52](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way, as described in "[Configuring One-Way SSL on WebLogic Server](#)"
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed using the Remote Console, as described in "[Supported Authentication Providers in WebLogic Server](#)".

17.57 oracle/http_basic_auth_over_ssl_service_policy

The oracle/http_basic_auth_over_ssl_service_policy uses the credentials in the HTTP header to authenticate users against the OPSS identity store and verifies that the transport protocol is HTTPS.

Display Name: HTTP Basic Auth Over SSL Service Policy

Category: Security

Description

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 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 Element](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_http_token_over_ssl_service_template](#)

The assertion is advertised in the WSDL.

Note:

Advertisement of policy assertions in a WADL file is not supported. The `Advertised` option has no effect when the associated policy is attached to a RESTful web service.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-53](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed using the Remote Console, as described in "[Supported Authentication Providers in WebLogic Server](#)".

17.58 oracle/http_mutual_auth_over_ssl_client_policy

The `oracle/http_mutual_auth_over_ssl_client_policy` includes credentials in the HTTP header for outbound client requests and verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused.

Display Name: HTTP Mutual Auth Over SSL Client Policy

Category: Security

Description

This policy can be enforced on any HTTP-based client endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/http_mutual_auth_over_ssl_client_template](#)

The assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [wss_http_token_over_ssl_client_template](#) Configuration Properties.
- Configure two-way SSL.
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed using the Remote Console.

See Also:

- [oracle/http_mutual_auth_over_ssl_client_template](#)
- [Table 18-52](#)
- [Overriding Policy Configuration Properties](#)
- [Configuring Two-Way SSL on WebLogic Server](#)
- [Supported Authentication Providers in WebLogic Server](#)

17.59 oracle/http_mutual_auth_over_ssl_service_policy

The `http_mutual_auth_over_ssl_service_policy` uses the credentials in the HTTP header to authenticate users against the OPSS identity store and verifies that the transport protocol is HTTPS.

Display Name: http mutual auth over ssl service policy

Category: Security

Description

Requests over a non-HTTPS transport protocol are refused. This policy can be enforced on any HTTP-based endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/http_mutual_auth_over_ssl_service_template](#)

The assertion is advertised in the WSDL.

 **Note:**

Advertisement of policy assertions in a WADL file is not supported. The Advertised option has no effect when the associated policy is attached to a RESTful web service.

Configuration

To configure the policy:

- Override the configuration properties defined in `wss_http_token_over_ssl_service_template` Configuration Properties.
- Configure two-way SSL.
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed using the Remote Console.

 **See Also:**

- [oracle/http_mutual_auth_over_ssl_service_template](#)
- [Table 18-53](#)
- [Overriding Policy Configuration Properties](#)
- [Configuring Two-Way SSL on WebLogic Server](#)
- [Supported Authentication Providers in WebLogic Server](#)

17.60 oracle/http_oam_token_service_policy

The `oracle/http_oam_token_service_policy` verifies that the OAM agent has authenticated the user and has established an identity.

Display Name: HTTP OAM Service Policy

Category: Security

Description

This policy can be enforced on any HTTP-based endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/http_oam_token_service_template](#)

The assertion is not advertised in the WSDL.

 **Note:**

Advertisement of policy assertions in a WSDL file is not supported. The Advertised option has no effect when the associated policy is attached to a RESTful web service.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-5](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- To enforce HTTP OAM security, configure OAM WebGate to intercept the request, authenticate the user, and set the `OAM_REMOTE_USER` HTTP header. OWSM verifies that the `OAM_REMOTE_USER_HTTP` header is present before allowing the request.
- To support remote user header, ensure that the `remote-user` configuration property value is set to the default value of `OAM_REMOTE_USER`.

For more information, see **Installing and Configuring Oracle HTTP Server 11g WebGate for OAM** in *Installing WebGates for Oracle Access Manager*.

17.61 oracle/http_saml20_token_bearer_client_policy

The `oracle/http_saml20_token_bearer_client_policy` includes a SAML Bearer V2.0 token in the HTTP header. The SAML token with confirmation method *Bearer* is created automatically.

Display Name: HTTP SAML Bearer V2.0 Token Client Policy

Category: Security

Description

This policy can be enforced on any HTTP-based client endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/http_saml20_token_bearer_client_template](#)

The assertion is advertised.

Configuration

To configure the policy, override the configuration properties defined in [Table 18-7](#). For more information, see "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

Configure SAML for the web service client at design time, as described in "[Configuring SAML Web Service Client at Design Time](#)".

17.62 oracle/http_saml20_token_bearer_service_policy

The oracle/http_saml20_token_bearer_service_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.

Display Name: HTTP Saml Bearer V2.0 Token Service Policy

Category: Security

Description

This policy can be enforced on any HTTP-based endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/http_saml20_token_bearer_service_template](#)

The assertion is advertised in the WSDL.

Note:

Advertisement of policy assertions in a WADL file is not supported. The Advertised option has no effect when this policy is attached to a RESTful web service.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-8](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure SAML and set up OPSS, as described in "[About SAML Configuration](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml2.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

17.63 oracle/http_saml20_token_bearer_over_ssl_client_policy

The oracle/http_saml20_token_bearer_over_ssl_client_policy includes a SAML Bearer v2.0 token in the HTTP header. The SAML token with confirmation method *Bearer* is created automatically, and verifies that the transport protocol provides SSL message protection.

Display Name: HTTP Saml Bearer V2.0 Token Over SSL Client Policy

Category: Security

Description

This policy can be attached to any HTTP-based client endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/http_saml20_token_bearer_client_template](#)

The assertion is advertised.

**Note:**

Advertisement of policy assertions in a WADL file is not supported. The Advertised option has no effect when this policy is attached to a RESTful web service.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-7](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way SSL, as described in "[Configuring One-Way SSL for a Web Service Client](#)".

Design Time Considerations

Configure SAML for the web service client at design time, as described in "[Configuring SAML Web Service Client at Design Time](#)".

17.64 oracle/http_saml20_bearer_token_over_ssl_service_policy

The oracle/http_saml20_bearer_token_over_ssl_service_policy authenticates users using credentials provided in the SAML v2.0 token with confirmation method *Bearer* in the HTTP header, and verifies that the transport protocol provides SSL message protection.

Display Name: HTTP Saml Bearer V2.0 Token Service Policy

Category: Security

Description

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/http_saml20_token_bearer_service_template](#)

The assertion is advertised in the WSDL.

 **Note:**

Advertisement of policy assertions in a WADL file is not supported. The Advertised option has no effect when this policy is attached to a RESTful web service.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-8](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure SAML and set up OPSS, as described in "[About SAML Configuration](#)".
- Configure one-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml2.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

17.65 oracle/multi_token_rest_service_policy

The oracle/multi_token_rest_service_policy enforces an authentication policy, based on the token sent by the client.

Display Name: Multi Token RESTful Service Policy

Category: Security

Description

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 v2.0 *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) token from the HTTP header.

Assertions (OR Group)

This policy contains assertions that are based on the following assertion templates as an OR group—meaning any one of the tokens can be sent by the client:

- [oracle/wss_http_token_client_template](#).
- [oracle/http_saml20_token_bearer_client_template](#).
- [oracle/http_oam_token_service_template](#). (Provides OAM protection on the server-side only.)

 **Note:**

For this policy, the default value of the `remote-user` configuration property is set to `NONE` to disable the processing of remote user header.

- [oracle/http_spnego_token_service_template](#).

The `oracle/http_saml20_token_bearer_client_template` and `oracle/http_spnego_token_service_template` policy assertions are advertised.

The `wss_http_token_client_template` and `oracle/http_oam_token_service_template` assertions are not advertised in the WSDL.

 **Note:**

Advertisement of policy assertions in a WADL file is not supported. The `Advertised` option has no effect when this policy is attached to a RESTful web service.

Configuration

To configure the policy:

- Override the configuration properties defined in one of the following sections, based on the token sent by the client. For more information, see "[Overriding Policy Configuration Properties](#)".
 - [Table 18-53](#)
 - [Table 18-8](#)
 - [Table 18-5](#)
 - [Table 18-11](#)
- To configure HTTP OAM security:
 - Configure the OAM service endpoint as `anonymous` using the OAM Console.
 - Configure OAM WebGate to intercept a client request, authenticate the user, and set the `OAM_REMOTE_USER` HTTP header. OWSM verifies that the `OAM_REMOTE_USER_HTTP` header is present before allowing the request.

For more information, see **Installing and Configuring Oracle HTTP Server 11g WebGate for OAM** in *Installing WebGates for Oracle Access Manager*.

17.66 oracle/multi_token_over_ssl_rest_service_policy

The `oracle/multi_token_over_ssl_rest_service_policy` enforces an authentication policy, based on the token sent by the client.

Display Name: Multi Token Over SSL RESTful Service Policy

Category: Configuration

Description

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.
- SPNEGO over HTTP security (non-SSL)—Extracts SPNEGO token information from the HTTP header.

Assertions (OR Group)

This policy contains assertions that are based on 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_client_template](#)
- [oracle/http_saml20_token_bearer_service_template](#)
- [oracle/http_oam_token_service_template](#)

 **Note:**

For this policy, the default value of the `remote-user` configuration property is set to `NONE` to disable the processing of remote user header.

- [oracle/http_spnego_token_service_template](#)

The `oracle/wss_http_token_over_ssl_client_template`, `oracle/http_saml20_token_bearer_service_template`, and `oracle/http_spnego_token_service_template` assertions are advertised in the WSDL.

The `oracle/http_oam_token_service_template` assertions are not advertised in the WSDL.

 **Note:**

Advertisement of policy assertions in a WADL file is not supported. The `Advertised` option has no effect when this policy is attached to a RESTful web service.

Configuration

To configure the policy:

- Override the configuration properties defined in one of the following sections, based on the token sent by the client. For more information, see "[Overriding Policy Configuration Properties](#)".
 - [Table 18-53](#)
 - [Table 18-8](#)
 - [Table 18-5](#)
 - [Table 18-11](#)
- To configure HTTP OAM security:

- Configure the OAM service endpoint as `anonymous` using the OAM Console.
- Configure OAM WebGate to intercept the request, authenticate the user, and set the `OAM_REMOTE_USER` HTTP header. OWSM verifies that the `OAM_REMOTE_USER_HTTP` header is present before allowing the request.

For more information, see **Installing and Configuring Oracle HTTP Server 11g WebGate for OAM** in *Installing WebGates for Oracle Access Manager*.

17.67 oracle/multi_token_sso_over_ssl_rest_service_policy

The `oracle/multi_token_sso_over_ssl_rest_service_policy` enforces an authentication policy, based on the token sent by the client.

Display Name: Multi Token SSO Over SSL RESTful Service Policy

Category: Security

Description

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 v2.0 *Bearer* token in the HTTP header—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

Assertions (OR Group)

This policy contains assertions that are based on 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](#)
- [oracle/http_saml20_token_over_ssl_bearer_service_policy](#)
- [oracle/http_oam_token_service_template](#) (Provides non-SSL OAM protection on the server-side only.)
- [oracle/http_spnego_token_service_template](#) (Provides non-SSL protection only.)
- [oracle/http_jwt_token_over_ssl_service_template](#)

Configuration

To configure the policy:

- To configure HTTP OAM security:

For more information, see **Installing and Configuring Oracle HTTP Server 11g WebGate for OAM** in *Installing WebGates for Oracle Access Manager*.

- To support remote user header, ensure that the `remote-user` configuration property value is set to the default value of `OAM_REMOTE_USER`.

17.68 oracle/multi_token_sso_rest_service_policy

The oracle/multi_token_sso_rest_service_policy enforces an authentication policy, based on the token sent by the client.

Display Name: Multi Token SSO Over SSL RESTful Service Policy

Category: Security

Description

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 v2.0 *Bearer* token in the HTTP header—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

Assertions (OR Group)

This policy contains assertions that are based on 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](#)
- [oracle/http_saml20_token_over_ssl_bearer_service_policy](#)
- [oracle/http_oam_token_service_template](#) (Provides non-SSL OAM protection on the server-side only.)
- [oracle/http_spnego_token_service_template](#) (Provides non-SSL protection only.)
- [oracle/http_jwt_token_over_ssl_service_template](#)

Configuration

To configure the policy:

- To configure HTTP OAM security:

For more information, see **Installing and Configuring Oracle HTTP Server 11g WebGate for OAM** in *Installing WebGates for Oracle Access Manager*.

- To support remote user header, ensure that the `remote-user` configuration property value is set to the default value of `OAM_REMOTE_USER`.

17.69 oracle/no_authentication_client_policy

The oracle/no_authentication_client_policy is a no behavior 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.

Display Name: No Behavior Authentication Client Policy

Category: Security

Description

If the globally attached policy contains any other assertions, in addition to the authentication assertion, those assertions are disabled as well. For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no_behavior policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-42](#) lists the configuration property that you can override for the no behavior policy.

Table 17-42 Configuration Property for oracle/no_authentication_client_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.70 oracle/no_authentication_service_policy

The oracle/no_authentication_service_policy is a no behavior 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.

Display Name: No Behavior Authentication Service Policy

Category: Security

Description

If the globally attached policy contains any other assertions, in addition to the authentication assertion, those assertions are disabled also. For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

 **Note:**

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no_behavior policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-43](#) lists the configuration property that you can override for the no behavior policy.

Table 17-43 Configuration Property for oracle/no_authentication_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.71 oracle/wss_http_token_client_policy

The oracle/wss_http_token_client_policy includes credentials in the HTTP header for outbound client requests. The client must pass the credentials in the HTTP header.

Display Name: Wss HTTP Token Client Policy

Category: Security

Description

This policy can be enforced on any HTTP-based client.

 **Note:**

Currently only HTTP basic authentication is supported.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_http_token_client_template](#)

This assertion is not advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-13](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- Specify a value for `csf-key`, as described in "[Overriding Policy Configuration Properties](#)". The value signifies a key that maps to a username/password. For information about how to add the key to the credential store, see "[Adding Keys and User Credentials to Configure the Credential Store](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- The client must pass the credentials in the HTTP header.

17.72 oracle/wss_http_token_service_policy

The `oracle/wss_http_token_service_policy` uses the credentials in the HTTP header to authenticate users against the OPSS identity store. This policy can be enforced on any HTTP-based endpoint.

Description

The web service must authenticate the supplied username and password credentials against the configured authentication source.



Note:

Currently only HTTP basic authentication is supported.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_http_token_service_template](#)

This assertion is not advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-14](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.

- The web service must authenticate the supplied username and password credentials against the configured authentication source. Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".

17.73 oracle/wss_username_token_client_policy

The oracle/wss_username_token_client_policy includes credentials in the WS-Security UsernameToken header for all outbound SOAP request messages. This policy can be attached to any SOAP-based client.

Display Name: Wss Username Token Client Policy

Category: Security

Description

To protect against replay attacks, the assertion provides the option to require nonce and 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 supports plain text passwords. This client policy is analogous to the oracle/wss_username_token_service_policy service endpoint policy.

Note:

This policy 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:

- Copying the policy and setting the password type to digest, as described in "[Creating and Editing Web Service Policies](#)".
- Using the SSL version of this policy, "[oracle/wss_username_token_over_ssl_client_policy](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_username_token_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-16](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Specify a value for `csf-key`, as described in "[Overriding Policy Configuration Properties](#)". The value signifies a key that maps to a username/password. For information about how to add the key to the credential store, see "[Adding Keys and User Credentials to Configure 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.

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Include a WS-Security UsernameToken element (<wsse:UsernameToken/>) in the SOAP request message. The client provides a username and password for authentication.

17.74 oracle/wss_username_token_service_policy

The oracle/wss_username_token_service_policy uses the credentials in the UsernameToken WS-Security SOAP header to authenticate users.

Display Name: Wss Username Token Service Policy

Category: Security

Description

This policy supports plain text passwords.

Note:

This policy 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:

- Copying the policy and setting the password type to digest, as described in "[Creating and Editing Web Service Policies](#)".
- Using the SSL version of this policy, "[oracle/wss_username_token_over_ssl_client_policy](#)".

To protect against replay attacks, the assertion provides the option to require nonce and 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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_username_token_service_template](#)

This assertion is not advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-17](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".

17.75 oracle/wss10_saml_token_client_policy

The oracle/wss10_saml_token_client_policy includes SAML tokens in outbound SOAP request messages.

Display Name: Wss10 SAML Token Client Policy

Category: Security

Description

The policy can be enforced on any SOAP-based client.

 **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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml_token_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-19](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure SAML and set up OPSS, as described in "[About SAML Configuration](#)".
- Specify a value for `saml.issuer.name`, as described in "[Overriding Policy Configuration Properties](#)". The `saml.issuer.name` property defaults to a value of `www.oracle.com`. For additional considerations, see "[Adding an Additional SAML Assertion Issuer Name](#)".
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure SAML for the web service client at design time, as described in "[Configuring SAML Web Service Client at Design Time](#)".
- 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*.

17.76 oracle/wss10_saml_token_service_policy

You can use the oracle/wss10_saml_token_service_policy to authenticate users using the credentials provided in SAML tokens in the WS-Security SOAP header. The credentials in the SAML token are authenticated against a SAML login module.

Display Name: Wss10 SAML Token Service Policy

Category: Security

Description

This policy can be enforced on any SOAP-based endpoint.



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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml_token_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-20](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure SAML and set up OPSS, as described in "[About SAML Configuration](#)".
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".

- Configure the `saml.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the provider.

17.77 oracle/wss10_saml20_token_client_policy

The `oracle/wss10_saml20_token_client_policy` includes SAML tokens in outbound SOAP request messages.

Display Name: Wss10 SAML V2.0 Token Client Policy

Category: Security

Description

The policy can be enforced on any SOAP-based client.



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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml20_token_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-22](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure SAML and set up OPSS, as described in "[About SAML Configuration](#)".
- Specify a value for `saml.issuer.name`, as described in "[Overriding Policy Configuration Properties](#)". The `saml.issuer.name` property defaults to a value of `www.oracle.com`. For additional considerations, see "[Adding an Additional SAML Assertion Issuer Name](#)".
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".

- Configure SAML for the web service client at design time, as described in "[Configuring SAML Web Service Client at Design Time](#)".
- 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*.

17.78 oracle/wss10_saml20_token_service_policy

The oracle/wss10_saml20_token_service_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.

Display Name: Wss10 SAML V2.0 Token Service Policy

Category: Security

Description

This policy can be enforced on any SOAP-based endpoint.

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml20_token_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-23](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure SAML and set up OPSS, as described in "[About SAML Configuration](#)".
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml2.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the provider.

17.79 oracle/wss11_kerberos_token_client_policy

The oracle/wss11_kerberos_token_client_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.

Display Name: Wss11 Kerberos Token Client Policy

Category: Security

Description

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_kerberos_token_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-25](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure Kerberos, as described in "[Understanding Kerberos Token Configuration](#)".
- 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`, as described in "[Overriding Policy Configuration Properties](#)". The default value (place holder) is `HOST/localhost@oracle.com`.

Design Time Considerations

At design time:

- Configure Kerberos, as described in "[Understanding Kerberos Token Configuration](#)".
- Set the service principal name (`service.principal.name`). The service principal name specifies the name of the service principal for which the client requests a ticket from the KDC. For more information, see "[Overriding Policy Configuration Properties](#)".
- 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.

17.80 oracle/wss11_kerberos_token_service_policy

The oracle/wss11_kerberos_token_service_policy extracts the Kerberos token from the SOAP header and authenticates the user. This policy is enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard. The container must have the Kerberos infrastructure configured through OPSS.

Display Name: Wss11 Kerberos Token Service Policy

Category: Security

Description

This policy is compatible with MIT and Active Directory KDCs. This policy can be attached to any SOAP-based endpoint.

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_kerberos_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-26](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the `krb5.loginmodule` login module, as described in "[Configuring the Kerberos Login Module](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".

17.81 oracle/http_oauth2_token_client_policy

You can use the oracle/http_oauth2_token_client_policy for attaching to any HTTP-based SOAP or REST client.

Display Name: Http OAuth2 Token Client Policy

Category: Security

Description

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.

Assertion

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](#)" for more information about the assertion.

Configuration

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](#)" for information about the assertion attributes that you can configure.

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>
</orawsp:http-oauth2-security>

```

Settings

See [Table 18-27](#).

Configuration Properties

See [Table 18-28](#).

17.82 oracle/ http_oauth2_token_with_resource_owner_creds_client_policy

The oracle/ http_oauth2_token_with_resource_owner_creds_client_policy includes the OAuth2 access token in the HTTP header. The access token (AT) is obtained from the Mobile & Social OAuth2 Server.

Display Name: Http OAuth2 token with resource owner creds client policy

Category: Security

Description

You can attach this policy to any HTTP-based client.

Assertion

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](#)".

Configuration

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.

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](#)" for information about the assertion attributes that you can configure.

You have to import the users from service domain to client domain as well as in the OAuth Server domain before you attach the policy.

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_with_resource_owner_creds_client_policy assertion content is defined as follows:

```
<?xml version = '1.0'?>
<wsp:Policy
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:oralgp="http://schemas.oracle.com/ws/2006/01/loggingpolicy" xmlns:orawsp="http://
schemas.oracle.com/ws/2006/01/policy" orawsp:provides="{http://docs.oasis-open.org/ns/
opencsa/sca/200912}authentication, {http://docs.oasis-open.org/ns/opencsa/sca/
200912}clientAuthentication, {http://schemas.oracle.com/ws/2006/01/policy}SOAP_HTTP,
{http://schemas.oracle.com/ws/2006/01/policy}REST_HTTP" orawsp:status="enabled"
xmlns="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.xsd"
wsu:Id="http_oauth2_token_with_resource_owner_creds_client_policy"
orawsp:displayName="i18n:oracle.wsm.resources.policydescription.PolicyDescriptionBundle_o
racle/http_oauth2_token_with_resource_owner_creds_client_policy_PolyDispNameKey"
xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
orawsp:description="i18n:oracle.wsm.resources.policydescription.PolicyDescriptionBundle_o
racle/http_oauth2_token_with_resource_owner_creds_client_policy_PolyDescKey"
orawsp:attachTo="binding.client" Name="oracle/
http_oauth2_token_with_resource_owner_creds_client_policy" orawsp:readOnly="true"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" orawsp:category="security"
orawsp:local-optimization="check-identity">
  <oralgp:Logging orawsp:Silent="true" orawsp:name="Log Message1"
orawsp:Enforced="false" orawsp:category="security/logging">
    <orlagp:msg-log>
      <oralgp:request>alloralgp:request>all>
      <oralgp:response>alloralgp:response>all>
      <oralgp:fault>alloralgp:fault>all>
    </oralgp:msg-log>
  </orawsp:bindings>
```

```

        <orawsp:Config orawsp:name="Log Message1_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>
</oralgp:Logging>
    <orasp:http-oauth2-security xmlns:ns0="http://schemas.oracle.com/ws/
2006/01/policy" ns0:Silent="false" ns0:name="Http OAuth2" ns0:Enforced="true"
ns0:category="security/authentication">
        <orasp:auth-header orasp:mechanism="oauth2"/>
        <orawsp:bindings>
            <orawsp:Config orawsp:name="HttpOAuth2Config"
orawsp:configType="declarative">
                <orawsp:PropertySet orawsp:name="standard-security-
properties">
                    <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="subject.precedence">
                        <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="constant" orawsp:name="grant_type">
                        <orawsp:DefaultValue>password</orawsp:DefaultValue>
                    </orawsp:Property>
                    <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="oauth2.client.csf.key">
                        <orawsp:Value/>
                    </orawsp:Property>
                    <orawsp:Property orawsp:type="boolean"
orawsp:contentType="optional" orawsp:name="federated.client.token">
                        <orawsp:Value/>
                    </orawsp:Property>
                    <orawsp:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="scope">
                        <orawsp:Value/>
                    </orawsp:Property>
                <!-- Begin : properties needed for local token creation
for end user -->
                    <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:Property>
                </orawsp:PropertySet>
            </orawsp:Config>
        </orawsp:bindings>
    </orasp:http-oauth2-security>
</orasp:auth-header>
</orasp:auth-header>

```

```

        </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:Property orawsp:type="string"
orawsp:contentType="optional" orawsp:name="token.lifetime">
            <orawsp:Value/>
        </orawsp:Property>
        <!--End properties for local token creation for end user --
>
        </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>
</oralgp:Logging>
</wsp:Policy>

```

Settings

See [Table 18-27](#).

Configuration Properties

See [Table 18-28](#).

17.83 oracle/ http_oauth2_token_with_resource_owner_creds_over_ssl_client _policy

The oracle/ http_oauth2_token_with_resource_owner_creds_over_ssl_client_policy includes the OAuth2 access token in the HTTP header. The access token (AT) is obtained from the Mobile & Social OAuth2 Server.

Display Name: Http OAuth2 token with resource owner creds over ssl client policy

Category: Security

Description

You can attach this policy to any HTTP-based client.

Assertion

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](#)".

Configuration

See [oracle/ http_oauth2_token_with_resource_owner_creds_client_policy](#).

Settings

See [Table 18-27](#).

Configuration Properties

See [Table 18-28](#).

17.84 oracle/http_jwt_token_service_policy

You can use the oracle/http_jwt_token_service_policy to authenticate users using the username provided in the JWT token in the HTTP header.

Display Name: Http Jwt Token Service Policy

Category: Security

Description

This policy can be applied to any HTTP-based endpoint.

Assertion

This policy contains the following policy assertion:

oracle/http_jwt_token_service_template

See "[oracle/http_jwt_token_service_template](#)" for more information about the assertion.

Configuration

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](#)" 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>
```

Settings

See [Table 18-37](#).

Configuration Properties

See [Table 18-29](#).

17.85 oracle/ http_oauth2_token_identity_switch_over_ssl_client_policy

The oracle/http_oauth2_token_identity_switch_over_ssl_client_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.

Display Name: Http OAuth2 Token Identity Switch Over Ssl Client Policy

Category: Security

Description

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 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.

Assertion

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](#)" for more information about the assertion.

Configuration

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.

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](#)" for information about the assertion attributes that you can configure.

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.

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>
</orawsp:http-oauth2-security>

```

Settings

See [Table 18-30](#).

Configuration Properties

See [Table 18-27](#).

17.86 oracle/http_jwt_token_over_ssl_service_policy

The oracle/http_jwt_token_over_ssl_service_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.

Display Name: HTTP JWT Token Over Ssl Service Policy

Category: Security

Description

This policy can be applied to any HTTP-based endpoint.

Assertion

This policy contains the following policy assertion:

oracle/http_jwt_token_over_ssl_service_template

See ["oracle/http_jwt_token_over_ssl_service_template"](#) for more information about the assertion.

Configuration

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"](#) 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>
```

Settings

See [Table 18-39](#).

Configuration Properties

See [Table 18-34](#).

17.87 oracle/http_oauth2_token_opc_oauth2_client_policy

The oracle/http_oauth2_token_opc_oauth2_client_policy includes the OAuth2 access token in the HTTP header. The access token is obtained from the Mobile & Social OAuth2 Server.

Display Name: HTTP OAuth2 Token Opc OAuth2 Client Policy

Category: Security

Description

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.

Assertion

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](#)" for more information about the assertion.

Configuration

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.

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](#)" for information about the assertion attributes that you can configure.

See "[Overriding Policy Configuration Properties](#)" 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>
</orawsp:http-oauth2-security>

```

Settings

See [Table 18-27](#).

Configuration Properties

See [Table 18-28](#).

17.88 oracle/http_oauth2_token_over_ssl_client_policy

The oracle/http_oauth2_token_over_ssl_client_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.

Display Name: HTTP OAuth2 Token Over SSL Client Policy

Category: Security

Description

The policy verifies that the outbound transport protocol is HTTPS. If a non-HTTPS transport protocol is used, the request is refused.

Assertion

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](#)" for more information about the assertion.

Configuration

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 OAuth2 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.

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](#)" for information about the assertion attributes that you can configure.

See "[Overriding Policy Configuration Properties](#)" 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"/>
<orasp:bindings>
<orasp:Config orasp:configType="declarative" orasp:name="HttpOAuth2OverSSLConfig">
<orasp:PropertySet orasp:name="standard-security-properties">
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="subject.precedence">
    <orasp:Value/>
    <orasp:DefaultValue>true</orasp:DefaultValue>
  </orasp:Property>
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="csf.map"/>
    <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="csf-key">
    <orasp:Value/>
  </orasp:Property>
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="oauth2.client.csf.key">
    <orasp:Value/>
    <orasp:DefaultValue>NONE</orasp:DefaultValue>
  </orasp:Property>
  <orasp:Property orasp:type="boolean" orasp:contentType="optional"
orasp:name="federated.client.token">
    <orasp:Value/>
    <orasp:DefaultValue>true</orasp:DefaultValue>
  </orasp:Property>
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="scope">
    <orasp:Value/>
  </orasp:Property>
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="authz.code">
    <orasp:Value/>
  </orasp:Property>
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="redirect.uri">
    <orasp:Value/>
  </orasp:Property>
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="user.attributes">
    <orasp:Value/>
  </orasp:Property>
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="issuer.name">
    <orasp:Value/>
    <orasp:DefaultValue>www.oracle.com</orasp:DefaultValue>
  </orasp:Property>
  <orasp:Property orasp:type="boolean" orasp:contentType="optional"
orasp:name="oracle.oauth2.service">
    <orasp:Value/>
    <orasp:DefaultValue>>false</orasp:DefaultValue>
  </orasp:Property>
  <orasp:Property orasp:type="boolean" orasp:contentType="optional"
orasp:name="user.roles.include">
    <orasp:Value/>
    <orasp:DefaultValue>>false</orasp:DefaultValue>
  </orasp:Property>
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="keystore.sig.csf.key">
    <orasp:Value/>
  </orasp: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>

```

Settings

See [Table 18-30](#).

Configuration Properties

See [Table 18-27](#).

17.89 oracle/http_jwt_token_over_ssl_service_policy

The oracle/http_jwt_token_over_ssl_service_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.

Display Name: HTTP Jwt Token Over SSL Service Policy

Category: Security

Description

This policy can be applied to any HTTP-based endpoint.

Assertion

This policy contains the following policy assertion: oracle/http_jwt_token_over_ssl_service_template. See "[oracle/http_jwt_token_over_ssl_service_template](#)" for more information about the assertion.

Configuration

For information about configuring the policy, see "[oracle/http_jwt_token_client_policy](#)".

17.90 oracle/oauth2_config_client_policy

The oracle/oauth2_config_client_policy provides OAuth2 information on the client side.

Display Name: OAuth2 Config Client Policy

Category: Security

Description

The OAuth2 information is used to invoke the Mobile and Social OAuth2 server for token exchange.

Assertion

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](#)" for more information about the assertion.

Configuration

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.

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](#)" for information about the assertion attributes that you can configure.

You must set or override the `token.uri` property. See "[Overriding Policy Configuration Properties](#)" 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>
```

Settings

See [Table 18-35](#).

Configuration Properties

See [Table 18-36](#).

17.91 oracle/http_jwt_token_client_policy

The `oracle/http_jwt_token_client_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.

Display Name: HTTP JWT Token Client Policy

Category: Security

Description

You can specify the audience restriction condition for this policy.

This policy can be enforced on any HTTP-based client endpoint.

Assertion

This policy contains the following policy assertion:

oracle/http_jwt_token_client_template

See "[oracle/http_jwt_token_client_template](#)" for more information about the assertion.

Configuration

The http_jwt_token_client_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 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](#)" 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 orasp:Enforced="true" orasp:Silent="false"
  orasp:category="security/authentication"
  orasp:name="Http JWT Security">
  <orasp:auth-header orasp:algorithm-suite="Basic128Sha256Rsa15"
    orasp:is-encrypted="false" orasp:is-signed="true" orasp:mechanism="jwt"/>
  <orasp:bindings>
    <orasp:Config orasp:configType="declarative"
      orasp:name="HttpJwtTokenConfig">
      <orasp:PropertySet orasp:name="standard-security-properties">
        <orasp:Property orasp:contentType="optional" orasp:name="user.attributes"
orasp:type="string"/>
        <orasp:Property orasp:contentType="optional" orasp:name="issuer.name"
orasp:type="string">
          <orasp:Value>www.oracle.com</orasp:Value>
        </orasp:Property>
        <orasp:Property orasp:contentType="optional"
orasp:name="user.roles.include" orasp:type="string">
          <orasp:Value>>false</orasp:Value>
        </orasp:Property>
        <orasp:Property orasp:contentType="optional" orasp:name="csf.map"
orasp:type="string"/>
        <orasp:Property orasp:contentType="optional" orasp:name="csf-key"
orasp:type="string">
          <orasp:Value>basic.credentials</orasp:Value>
        </orasp:Property>
        <orasp:Property orasp:contentType="optional"
orasp:name="subject.precedence" orasp:type="string">
          <orasp:Value>>true</orasp:Value>
        </orasp:Property>
        <orasp:Property orasp:contentType="optional" orasp:name="audience.uri"
orasp:type="string">
          <orasp:Value/>
        </orasp:Property>
        <orasp:Property orasp:contentType="optional"
orasp:name="keystore.sig.csf.key" orasp:type="string">
          <orasp:Value/>
        </orasp:Property>
        <orasp:Property orasp:contentType="optional"
orasp:name="propagate.identity.context" orasp: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>
</orawsp:http-jwt-security>

```

Settings

See [Table 18-37](#).

Configuration Properties

See [Table 18-38](#).

17.92 oracle/http_jwt_token_over_ssl_client_policy

The oracle/http_jwt_token_over_ssl_client_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.

Dsisplay Name: HTTP JWT Token Over SSL Client Policy

Category: Security

Description

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.

Assertion

This policy contains the following policy assertion: oracle/http_jwt_token_over_ssl_client_template. See "[oracle/http_jwt_token_over_ssl_client_template](#)" for more information about the assertion.

Configuration

For information about configuring the policy, see "[oracle/http_jwt_token_client_policy](#)".

17.93 oracle/http_oauth2_token_identity_switch_opc_oauth2_over_ssl_client_policy

The oracle/http_oauth2_token_identity_switch_opc_oauth2_over_ssl_client_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.

Display Name: HTTP OAuth2 Token Identity Switch Opc OAuth2 Over SSL Client Policy

Category: Security

Description

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.

Assertion

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](#)" for more information about the assertion.

Configuration

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.

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](#)" for information about the assertion attributes that you can configure.

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 "[Setting the Permission Using WSIdentityPermission](#)".

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>
</orawsp:http-oauth2-security>

```

Settings

See [Table 18-30](#).

Configuration Properties

See [Table 18-27](#).

17.94 oracle/ http_oauth2_token_opc_oauth2_over_ssl_client_policy

The oracle/http_oauth2_token_opc_oauth2_over_ssl_client_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.

Display Name: HTTP OAuth2 Token Opc OAuth2 Over SSL Client Policy

Category: Security

Description

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.

Assertion

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](#)" for more information about the assertion.

Configuration

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.

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](#)" for information about the assertion attributes that you can configure.

See "[Overriding Policy Configuration Properties](#)" 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>
```

Settings

See [Table 18-30](#).

Configuration Properties

See [Table 18-27](#).

17.95 oracle/http_jwt_token_identity_switch_client_policy

The 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.

Display Name: HTTP JWT Token Identity Switch Client Policy

Category: Security

Description

This policy can be enforced on any HTTP-based, SOAP, or REST client endpoint.

Assertion

This policy contains the following policy assertion:

oracle/http_jwt_token_client_template

See "[oracle/http_jwt_token_client_template](#)" for more information about the assertion.

Configuration

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](#)" 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_template](#)", 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>
```


Settings

See [Table 18-37](#).

Configuration Properties

See [Table 18-38](#).

17.96 oracle/binding_authorization_denyall_policy

The oracle/binding_authorization_denyall_policy provides a simple role-based authorization policy based on the authenticated Subject at the SOAP binding level.

Display Name: Binding Authorization DenyAll Policy

Category: Security

Description

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/binding_authorization_template](#)

This assertion is not advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-123](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- To set up OPSS:
 - If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. Use the WebLogic Server Remote Console to grant a role to a user or group, as described in the Manage users and groups.
 - Configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*.

17.97 oracle/binding_authorization_permitall_policy

The oracle/binding_authorization_permitall_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.

Display Name: Binding Authorization PermitAll Policy

Category: Security

Description

It should follow an authentication policy where the Subject is established and can be attached to any SOAP-based endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/binding_authorization_template](#)

This assertion is not advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-123](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- To set up OPSS:
 - If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. Use the WebLogic Server Remote Console to grant a role to a user or group, as described in the Manage users and groups.
 - Configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*.

17.98 oracle/binding_permission_authorization_policy

The oracle/binding_permission_authorization_policy provides a permission-based authorization policy based on the authenticated subject. This policy should follow an authentication policy where the Subject is established and can be attached to any SOAP-based endpoint.

Display Name: Binding Permission Based Authorization Policy

Category: Security

Description

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. For more information, see "[Determining Authorization Permissions](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/binding_permission_authorization_template](#)

This assertion is not advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-125](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- To set up 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.
 - Optionally, 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 custom permission class must extend the abstract `Permission` class and implement the `Serializable` interface. See the Javadoc at <http://docs.oracle.com/javase/7/docs/api/java/security/Permission.html>. The default is `oracle.wsm.security.WSFunctionPermission`.
 - Configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*.

17.99 oracle/component_authorization_denyall_policy

The `oracle/component_authorization_denyall_policy` provides a simple role-based authorization policy based on the authenticated subject.

Display Name: Component Authorization DenyAll Policy

Category: Security

Description

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/component_authorization_template](#)

This assertion is not advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-127](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- To set up OPSS:
 - If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. Use the WebLogic Server Remote Console to grant a role to a user or group, as described in the Manage users and groups.

- Configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*.

17.100 oracle/component_authorization_permitall_policy

The oracle/component_authorization_permitall_policy provides a simple role-based authorization policy based on the authenticated subject.

Display Name: Component Authorization PermitAll Policy

Category: Security

Description

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/component_authorization_template](#)

This assertion is not advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-127](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- To set up OPSS:
 - If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. Use the WebLogic Server Remote Console to grant a role to a user or group, as described in the Manage users and groups.
 - Configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*.

17.101 oracle/component_permission_authorization_policy

The oracle/component_permission_authorization_policy provides a permission-based authorization policy based on the authenticated Subject. This policy should follow an authentication policy where the Subject is established and can be attached to any SCA-based endpoint.

Display Name: Component Permission Based Authorization Policy

Category: Security

Description

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`. For more information, see "[Determining Authorization Permissions](#)".

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/component_permission_authorization_template](#)

This assertion is not advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-129](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- To set up 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.
 - Configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*.

17.102 oracle/no_authorization_component_policy

The `oracle/no_authorization_component_policy` is a no behavior 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.

Display Name: No Behavior Authorization Component Policy

Category: Security

Description

If the globally attached policy contains any other assertions, in addition to the authorization assertion, those assertions are disabled as well. For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

 **Note:**

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no_behavior policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-44](#) lists the configuration property that you can override for the no behavior policy.

Table 17-44 Configuration Property for oracle/no_authorization_component_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.103 oracle/no_authorization_service_policy

The oracle/no_authorization_service_policy is a no behavior 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.

Display Name: No Behavior Authorization Service Policy

Category: Security

Description

If the globally attached policy contains any other assertions, in addition to the authorization assertion, those assertions are disabled also. For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

 **Note:**

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no_behavior policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-45](#) lists the configuration property that you can override for the no behavior policy.

Table 17-45 Configuration Property for oracle/no_authorization_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.104 oracle/whitelist_authorization_policy

The oracle/whitelist_authorization_policy is a special case of role based authorization policy, and accepts requests only if a specified condition is true.

Display Name: Constraints Based Authorization Policy

Category: Security

Description

This policy is a special case of role based authorization policy. This policy can be attached to any SOAP-based endpoint.

Accepts requests 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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/binding_authorization_template](#)

This assertion is not advertised in the WSDL.

Configuration

To configure the 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 the Oracle HTTP Server to Specify the Request Origin](#)"), then a client on the internal network only has to attach the corresponding username token client policy at the client side.
- To set up OPSS:
 - If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. Use the WebLogic Server Remote Console to grant a role to a user or group, as described in the Manage users and groups.
 - You must configure a WebLogic Authentication provider, as described in "Configure Authentication and Identity Assertion providers" in the *Oracle WebLogic Server Administration Console Online Help*.
 - 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 configure the Oracle HTTP Server, see "[Configuring the Oracle HTTP Server to Specify the Request Origin](#)".

17.105 oracle/no_messageprotection_client_policy

The `oracle/no_messageprotection_client_policy` is a no behavior 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.

Display Name: No Behavior Message Protection Client Policy

Category: Security

Description

If the globally attached policy contains any other assertions, in addition to the message protection assertion, those assertions are disabled also. For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This `no_behavior` policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-46](#) lists the configuration property that you can override for the no behavior policy.

Table 17-46 Configuration Property for oracle/no_messageprotection_client_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.106 oracle/no_messageprotection_service_policy

The oracle/no_messageprotection_service_policy, is a no behavior 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.

Display Name: No Behavior Message Protection Service Policy

Category: Security

Description

If the globally attached policy contains any other assertions, in addition to the message protection assertion, those assertions are disabled also. For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

**Note:**

This policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-47](#) lists the configuration property that you can override for the no behavior policy.

Table 17-47 Configuration Property for oracle/no_messageprotection_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.107 oracle/wss10_message_protection_client_policy

The oracle/wss10_message_protection_client_policy provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

Display Name: Wss10 Message Protection Client Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-42](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- To configure OPSS, set up the OWSM keystore and the web service client keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)". 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's base64-encoded public certificate is published in the WSDL for use by the web service client, as described in "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".
- Configure the policy assertion for message signing, message encryption, or both.

Design Time Considerations

At design time:

- Override configuration settings, as described in "About Overriding Client Policy Configuration Properties at Design Time".
- Configure the policy assertion for message signing, message encryption, or both.
- You can include signature and encryption elements in the Security header in conformance with the WS-Security 1.0 standards.

The following example (WS-Security 1.0 Message Integrity of SOAP Message) shows the typical structure of a signature included in the Security header. In this example, the body element of the SOAP message is signed.

```
<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-
security-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-profile-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>
```

The following example (WS-Security 1.0 Message Confidentiality of SOAP Message) shows 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.

```

<env:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-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>

```

17.108 oracle/wss10_message_protection_service_policy

The oracle/wss10_message_protection_service_policy enforces message protection (integrity and confidentiality) for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

Display Name: Wss10 Message Protection Service Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-43](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
 - 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.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

17.109 oracle/wss11_message_protection_client_policy

The oracle/wss11_message_protection_client_policy provides message integrity and confidentiality for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

Display Name: Wss11 Message Protection Client Policy

Category: Security

Description

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](#)".

Symmetric key technology is an encryption method that uses the same shared key to encrypt and decrypt data. The symmetric key is used to sign the message.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-45](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Set up the OWSM keystore and the web service client keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)". 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's base64-encoded public certificate is published in the WSDL for use by the web service client, as described in "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Configure the policy assertion for message signing, message encryption, or both.
- Specify a value for `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. 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.
- Configure the policy assertion for message signing, message encryption, or both.

The following example (WS-Security 1.1 Message Confidentiality of SOAP Message) shows 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.

```
<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#ThumbprintSHA1"
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-
utility-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-secext-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#EncryptedKey" />
      </wsse:SecurityTokenReference>
    </dsig:KeyInfo>
    <xenc:CipherData>
      <xenc:CipherValue>...</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</env:Body>
```

17.110 oracle/wss11_message_protection_service_policy

The oracle/wss11_message_protection_service_policy enforces message integrity and confidentiality for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

Display Name: Wss11 Message Protection Service Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-46](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- To set up OPSS:
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
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.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Configure the policy assertion for message signing, message encryption, or both.
 - Specify a value for `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

17.111

wss11_username_token_derivedkey_with_message_protection_service

The `oracle/wss11_username_token_derivedkey_with_message_protection_service_policy` enables use of OWSM to integrate with client where request contains `<wsse11:Salt>` or `<wsse11:Iteration>` element in the username token. These elements are used in Username token to facilitate password-derived keys support. Either signature or encryption is used.

Display Name: Wss11 Username Token With Message Protection using Password Derived Keys Service Policy

Category: Security

Description

The web service consumer inserts username and password credentials, and signs or encrypts the outgoing SOAP message. The web service provider decrypts or verifies the message. This policy can be attached to any SOAP-based endpoint.

To protect against replay attacks, the assertion provides the option to require nonce and creation time in the username token.

To prevent replay attacks, the assertion provides the option to include time stamps and verification by the web service provider.

This policy uses the WS-Security Basic 128 Algorithm suite symmetric key technology for message signing or encryption. See [Web Services Security UsernameTokenProfile 1.1 — OASIS Public Review Draft - 28 June 2005](#).



Note:

Only BASIC128 ALgosuite is supported for this policy.

The steps for key derivation for each request include:

1. The client creates a **secret key** using the password associated with the user. This is used to create a symmetric signature or encryption of data according to the applied client policy.



Note:

The UsernameToken header encryption is not supported.

2. When the service receives the message, it derives the same secret key as the client using its knowledge of the password and two additional elements, that is, salt and iteration as received in the request token.
3. The Web service authenticates the user passed through the UsernameToken and decrypts or verifies the message using this password derived key.
4. It then uses the same secret key to encrypt or sign the response that it sends back to the client.

Assertion (OR Group)

This service policy contains assertions for signature and encryption. Based on the type of assertion used by the client policy, either signature or encryption is used. The client policy templates are:

- [wss11_username_token_derivedkey_with_message_protection_signature_only_client_template](#)
- [wss11_username_token_derivedkey_with_message_protection_encryption_only_client_template](#)

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-102](#). For more information, see [Overriding Policy Configuration Properties](#).

- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in [Supported Authentication Providers in WebLogic Server](#).
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in [Overview of Configuring Keystores for Message Protection](#).
- Set the value of `user.csf.key` configuration parameter. The default value is `basic.credentials`.

17.112 oracle/wss11_username_token_with_message_protection_client_policy

The `oracle/wss11_username_token_with_message_protection_client_policy` enables use of OWSM to integrate with any backend service which requires `<wsse11:Salt>` or `<wsse11:Iteration>` element in the username token. These elements are used in Username token to facilitate password-derived keys support. This client policy is for message protection using signature.

Display Name: Wss11 Username Token With Message Protection Signature using Password Derived Keys Client Policy

Category: Security

Description

The web service consumer inserts username and password credentials, and signs the outgoing SOAP message. The web service provider then verifies the message signature.

To protect against replay attacks, the assertion provides the option to require nonce and creation time in the username token. The SOAP message is signed. The web service provider verifies and authenticates the signature.

To prevent replay attacks, the assertion provides the option to include time stamps and verification by the web service provider.

This policy uses the WS-Security's Basic 128 suite of symmetric key technology for message signing.



Note:

Only BASIC128 ALGosuite is supported for this policy.

The steps for key derivation for each request include:

1. The client creates a **secret key** using the password associated with the user. This secret key is used to create a symmetric signature of data.
2. When the service receives the message, it derives the same secret key as the client using its knowledge of the password and two additional elements, that is, salt and iteration as received in the request token.

3. The Web service authenticates the user passed through the UsernameToken and verifies the message using this password derived key.
4. It then uses the same secret key to sign the response that it sends back to the client.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_username_token_derivedkey_with_message_protection_signature_only_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-101](#). For more information, see [Overriding Policy Configuration Properties](#).
- Set up the OWSM keystore to specify a key (username/password).
- Configure the policy assertion for message signing.

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure the policy assertion for message signing.

17.113

wss11_username_token_derivedkey_message_protection_encryption_client

The `oracle/wss11_username_token_derivedkey_with_message_protection_encryption_only_client_policy` enables use of OWSM to integrate with any backend service which requires `<wss11:Salt>` or `<wss11:Iteration>` element in the username token. These elements are used in Username token to facilitate password-derived keys support. This client policy is for message protection using encryption.

Display Name: Wss11 Username Token With Message Protection Encryption using Password Derived Keys Client Policy

Category: Security

Description

The web service consumer inserts username and password credentials, and encrypts the outgoing SOAP message. The web service provider decrypts the message.

To protect against replay attacks, the assertion provides the option to require nonce and creation time in the username token. The SOAP message is encrypted. The web service provider decrypts the message, and authenticates the user.

To prevent replay attacks, the assertion provides the option to include time stamps and verification by the web service provider.

This policy uses the WS-Security's Basic 128 suite of symmetric key technology for encryption.

 **Note:**

Only BASIC128 ALgosuite is supported for this policy.

The steps for key derivation for each request include:

1. The client creates a secret key using the password associated with the user. This secret key is used for encryption.

 **Note:**

The UsernameToken header encryption is not supported.

2. When the service receives the message, it derives the same secret key as the client using its knowledge of the password and two additional elements, that is, salt and iteration as received in the request token.
3. The Web service authenticates the user passed through the UsernameToken and the message using this password derived key.
4. It then uses the same secret key to encrypt the response that it sends back to the client.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_username_token_derivedkey_with_message_protection_encryption_only_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-101](#). For more information, see [Overriding Policy Configuration Properties](#).
- Set up the OWSM keystore to specify a key (username/password)
- Configure the policy assertion for message encryption.

Design Time Considerations

At design time:

- Override configuration settings, as described in [About Overriding Client Policy Configuration Properties at Design Time](#).
- Configure the policy assertion for message encryption.

17.114 oracle/pii_security_policy

The oracle/pii_security_policy encrypts the Personally Identifiable Information (PII) data you want to protect.

Display Name: PII Security Policy

Category: Security

Description

Encrypts the Personally Identifiable Information (PII) data you want to protect.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/pii_security_template](#)

This assertion is not advertised in the WSDL.

Configuration

Override the configuration properties defined in [Table 18-109](#). For more information, see "[Overriding Policy Configuration Properties](#)".

17.115 oracle/sts_trust_config_client_policy

The oracle/sts_trust_config_client_policy specifies the STS client configuration information that is used to invoke the STS for token exchange.

Display Name: STS Trust Configuration Client Policy

Category: Security

Description

Use this policy only if you are not using Automatic (Client STS) Policy Configuration, as described in "[Setting Up Automatic Policy Configuration for STS](#)"

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/sts_trust_config_client_template](#)

This assertion is not advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-111](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the STS configuration policy from the web service, as described in "[Setting Up Automatic Policy Configuration for STS](#)".
- However, if you did not configure the STS configuration policy from the web service, or if you are using the SAML sender vouches confirmation method, then you must configure it from the web service client. For more information, see "[Manually Configuring the STS Config Policy From the Web Service Client: Main Steps](#)".

Design Time Considerations

At design time, you can set up and attach the `oracle/sts_trust_config_client_policy` policy programmatically, as shown in the following example.

```
URL endpointUrl = new URL(getWebConnectionString() + "/jaxws-test-service/jaxws-test-port");

ServiceDelegateImpl client = new ServiceDelegateImpl(
    new URL(endpointUrl.toString() + "?WSDL"),
    new QName("http://jaxws.example.com/targetNamespace/JaxwsService", "JaxwsService"),
    OracleService.class);

JaxwsService port = client.getPort(
    new QName("http://jaxws.example.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 following example shows the related `oracle-webservice-client.xml` file with the STS config policy and STS issue policy.

```
<?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>
```

17.116 oracle/sts_trust_config_service_policy

The oracle/sts_trust_config_service_policy specifies the STS configuration information that is used to invoke the STS for token exchange.

Display Name: STS Trust Configuration Service Policy

Category: Security

Description

Specifies the STS configuration information that is used to invoke the STS for token exchange.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/sts_trust_config_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-113](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Set up the web service, as described in "[Setting Up Automatic Policy Configuration for STS](#)".

17.117 oracle/wss_saml_bearer_or_username_token_service_policy

The oracle/wss_saml_bearer_or_username_token_service_policy enforces one authentication policy, based on whether the client uses a SAML or username token.

Display Name: WSSecurity SAML Token Bearer or WSSecurity UserName Token

Category: Security

Description

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.

To protect against replay attacks, the assertion provides the option to require nonce and 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.

Assertions (OR Group)

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_service_template](#)
- [oracle/wss_username_token_service_template](#)

The assertions are advertised in the WSDL.

17.118 oracle/wss_saml_or_username_token_service_policy

The oracle/wss_saml_or_username_token_service_policy enforces an authentication policy, based on whether the client uses a SAML or username token.

Display Name: Wss SAML Token or Wss Username Token Service Policy

Category: Security

Description

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.

To protect against replay attacks, the assertion provides the option to require nonce and 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.

Assertions (OR Group)

This policy contains an assertion that is based on the following assertion templates, as an OR group—meaning either one of the tokens can be sent by the client:

- [oracle/wss10_saml_token_service_template](#)
- [oracle/wss_username_token_service_template](#)

The assertions are advertised in the WSDL.

Configuration

For information about configuring this policy, refer to the following policy descriptions:

- ["oracle/wss10_saml_token_service_policy"](#)
- ["oracle/wss_username_token_service_policy"](#).

17.119 oracle/wss_saml_or_username_token_over_ssl_service_policy

The oracle/wss_saml_or_username_token_over_ssl_service_policy enforces message protection (integrity and confidentiality) and an authentication policy, based on whether the client uses a SAML or username token.

Display Name: Wss SAML Token or Wss Username Token Over SSL Service Policy

Category: Security

Description

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.

To protect against replay attacks, the assertion provides the option to require nonce and 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.

Assertions (OR Group)

This policy contains an assertion that is based on the following assertion templates as an OR group—meaning either one of the tokens can be sent by the client:

- [oracle/wss_saml_token_over_ssl_service_template](#)
- [oracle/wss_username_token_over_ssl_client_template](#)

The assertions are advertised in the WSDL.

Configuration

For information about configuring this policy, refer to the following policy descriptions:

- "oracle/wss_saml_token_over_ssl_service_policy"
- "oracle/wss_username_token_over_ssl_service_policy"

17.120 oracle/wss_saml_token_bearer_client_policy

The oracle/wss_saml_token_bearer_client_policy includes SAML tokens in outbound SOAP request messages.

Display Name: Wss SAML Token (confirmation method as bearer) Client Policy

Category: Security

Description

The SAML token with confirmation method *Bearer* is created automatically.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml_token_bearer_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-59](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure SAML on the client side, as described in "[Configuring SAML Web Service Client at Design Time](#)".

17.121 oracle/wss_saml_token_bearer_over_ssl_client_policy

The oracle/wss_saml_token_bearer_over_ssl_client_policy includes SAML tokens in outbound SOAP request messages. The policy also verifies that the transport protocol provides SSL message protection.

Display Name: Wss SAML Token (confirmation method as bearer) Over SSL Client Policy

Category: Security

Description

The SAML token with confirmation method *Bearer* is created automatically. This policy can be attached to any SOAP-based client.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml_token_bearer_over_ssl_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- This policy uses the sender vouches confirmation with message protection using SSL, therefore the issuer key identifier used in SSL certificate must be trusted. Configure trusted issuers and DN lists, as described in [Configuring SAML and JWT Trusted Issuers, DN Lists, and Token Attribute Rules Using WLST](#)

 **Note:**

You do not need to configure trusted issuers and DN lists, If the DN is not configured for the issuer or you are using the default trusted SAML issuer (www.oracle.com) for token issuer trust document.

- Override the configuration properties defined in [Table 18-59](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure SAML on the client side, as described in "[Configuring SAML Web Service Client at Design Time](#)".

17.122 oracle/wss_saml_token_bearer_over_ssl_service_policy

The `oracle/wss_saml_token_bearer_over_ssl_service_policy` authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.

Display Name: Wss SAML Token (confirmation method as bearer) Over SSL Service Policy

Category: Security

Description

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.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml_token_bearer_over_ssl_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-60](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml.loginmodule` login module. See "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)" for more information. The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- Configure SAML and set up OPSS, as described in "[About SAML Configuration](#)".

17.123 oracle/wss_http_token_over_ssl_client_policy

The `oracle/wss_http_token_over_ssl_client_policy` includes credentials in the HTTP header for outbound client requests, authenticates users against the OPSS identity store, and verifies that the transport protocol is HTTPS. The client must pass the credentials in the HTTP header.

Display Name: Wss HTTP Token Over SSL Client Policy

Category: Security

Description

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_http_token_over_ssl_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-52](#). For more information, see "[Overriding Policy Configuration Properties](#)".

- Configure one-way SSL, as described in "[Configuring One-Way SSL for a Web Service Client](#)".
- Specify a value for `csf-key`, as described in "[Overriding Policy Configuration Properties](#)". The value signifies a key that maps to a username/password. For information about how to add the key to the credential store, see "[Adding Keys and User Credentials to Configure the Credential Store](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- The client must pass the credentials in the HTTP header.

17.124 oracle/wss_http_token_over_ssl_service_policy

The `oracle/wss_http_token_over_ssl_service_policy` extracts the credentials in the HTTP header and authenticates users against the OPSS identity store, and verifies that the transport protocol is HTTPS.

Display Name: Wss HTTP Token Over SSL Service Policy

Category: Security

Description

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/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 Element](#)".

Currently only HTTP basic authentication is supported.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_http_token_over_ssl_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-53](#). For more information, see "[Overriding Policy Configuration Properties](#)".

- Configure one-way SSL, as described in "[Configuring One-Way SSL for a Web Service Client](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".

17.125 oracle/wss_saml_token_over_ssl_client_policy

The oracle/wss_saml_token_over_ssl_client_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.

Display Name: Wss SAML Token Over SSL Client Policy

Category: Security

Description

This policy can be enforced on any SOAP-based client.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml_token_over_ssl_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- This policy uses the sender vouches confirmation with message protection using SSL, therefore the issuer key identifier used in SSL certificate must be trusted. Configure trusted issuers and DN lists, as described in [Configuring SAML and JWT Trusted Issuers, DN Lists, and Token Attribute Rules Using WLST](#).

 **Note:**

You do not need to configure trusted issuers and DN lists, If the DN is not configured for the issuer or you are using the default trusted SAML issuer (www.oracle.com) for token issuer trust document.

- Override the configuration properties defined in [Table 18-65](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure SAML on the client side, as described in "[Configuring SAML Web Service Client at Design Time](#)".

17.126 oracle/wss_saml_token_over_ssl_service_policy

The oracle/wss_saml_token_over_ssl_service_policy enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type, and verifies that the transport protocol provides SSL message protection.

Display Name: Wss SAML Token Over SSL Service Policy

Category: Security

Description

The SAML token is mapped to a user in the configured identity store. This policy can be enforced on any SOAP-based endpoint.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml_token_over_ssl_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-66](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the provider.
- Configure SAML and set up OPSS, as described in "[About SAML Configuration](#)".

17.127 oracle/wss_saml20_token_bearer_over_ssl_client_policy

The oracle/wss_saml20_token_bearer_over_ssl_client_policy includes SAML tokens in outbound SOAP request messages, and verifies that the transport protocol provides SSL message protection.

Display Name: Wss SAML V2.0 Token (confirmation method as bearer) Over SSL Client Policy

Category: Security

Description

The SAML token with confirmation method *Bearer* is created automatically. This policy can be attached to any SOAP-based client.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml20_token_bearer_over_ssl_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-62](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- 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 Using SAML Policies](#)" for additional considerations.

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure SAML on the client side, as described in "[Configuring SAML Web Service Client at Design Time](#)".

17.128 oracle/wss_saml20_token_bearer_over_ssl_service_policy

The oracle/wss_saml20_token_bearer_over_ssl_service_policy authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header, and verifies that the transport protocol provides SSL message protection.

Display Name: Wss SAML V2.0 Token (confirmation method as bearer) Over SSL Service Policy

Category: Security

Description

The credentials in the SAML token are authenticated against a SAML login module. This policy can be enforced on any SOAP-based endpoint.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml20_token_bearer_over_ssl_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-63](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml2.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the provider.
- Configure SAML and set up OPSS, as described in "[About SAML Configuration](#)".

17.129 oracle/wss_saml20_token_over_ssl_client_policy

The `oracle/wss_saml20_token_over_ssl_client_policy` includes SAML tokens in outbound WS-Security SOAP headers using the sender-vouches confirmation type, and verifies that the transport protocol provides SSL message protection.

Display Name: Wss SAML V2.0 Token Over SSL Client Policy

Category: Security

Description

This policy can be enforced on any SOAP-based client.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml20_token_over_ssl_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-68](#). For more information, see ["Overriding Policy Configuration Properties"](#).
- Configure one-way or two-way SSL, as described in ["Configuring One-Way SSL on WebLogic Server"](#) or ["Configuring Two-Way SSL for a Web Service Client"](#), respectively.
- Specify a value for `propagate.identity.context`, as described in ["Overriding Policy Configuration Properties"](#). The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see ["Propagating Identity Context Using SAML Policies"](#).

Design Time Considerations

At design time:

- Override configuration settings, as described in ["About Overriding Client Policy Configuration Properties at Design Time"](#).
- Configure SAML on the client side, as described in ["Configuring SAML Web Service Client at Design Time"](#).

17.130 oracle/wss_saml20_token_over_ssl_service_policy

The `oracle/wss_saml20_token_over_ssl_service_policy` enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type, and verifies that the transport protocol provides SSL message protection.

Display Name: Wss SAML V2.0 Token Over SSL Service Policy

Category: Security

Description

The SAML token is mapped to a user in the configured identity store. This policy can be enforced on any SOAP-based endpoint.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml20_token_over_ssl_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-63](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml2.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the provider.
- Configure SAML and set up OPSS, as described in "[About SAML Configuration](#)".

17.131 oracle/ wss_sts_issued_saml_bearer_token_over_ssl_client_policy

The oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_policy inserts a SAML bearer assertion issued by a trusted STS. Messages are protected using SSL.

Display Name: Wss Issued Token with Saml Bearer Over SSL Client Policy

Category: Security

Description

Inserts a SAML bearer assertion issued by a trusted STS. Messages are protected using SSL.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-115](#). For more information, see "[Overriding Policy Configuration Properties](#)". For examples of overriding STS configuration settings, see "[Programmatically Overriding Policy Configuration for WS-Trust Client Policies](#)".
- Set up the web service client, as described in "[Main Steps in Setting Up Automatic Policy Configuration](#)".
- Set up the OWSM keystore to specify a key (username/password or X.509) to authenticate to the STS, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure SAML on the client side, as described in "[Configuring SAML Web Service Client at Design Time](#)".

17.132 oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy

The oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_policy authenticates a SAML bearer assertion issued by a trusted STS.

Display Name: Wss Issued Token with Saml Bearer Over SSL Service Policy

Category: Security

Description

Authenticates a SAML bearer assertion issued by a trusted STS. Messages are protected using SSL.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template](#)

This assertion is advertised in the WSDL.

See also "[WS-Trust Assertion Templates](#)" for more information about the assertion.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-116](#). For more information, see "[Overriding Policy Configuration Properties](#)". For examples of overriding STS configuration settings, see "[Programmatically Overriding Policy Configuration for WS-Trust Client Policies](#)".
- Set up the web service, as described in "[Main Steps in Setting Up Automatic Policy Configuration](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.

17.133 oracle/wss_username_token_over_ssl_client_policy

The oracle/wss_username_token_over_ssl_client_policy includes credentials in the WS-Security UsernameToken header in outbound SOAP request messages, and verifies that the

transport protocol provides SSL message protection. Both plain text and digest mechanisms are supported.

Display Name: Wss Username Token Over SSL Client Policy

Category: Security

Description

This policy can be attached to any SOAP-based client.

To protect against replay attacks, the assertion provides the option to require nonce and 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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_username_token_over_ssl_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-71](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- If you specify a password type of None on the **Settings** page, you do not need to include a password in the key.
- Specify a value for `csf-key`, as described in "[Overriding Policy Configuration Properties](#)". The value signifies a key that maps to a username/password. For information about how to add the key to the credential store, see "[Adding Keys and User Credentials to Configure the Credential Store](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Include a WS-Security UsernameToken element (`<wsse:UsernameToken/>`) in the SOAP request message. The client provides a username and password for authentication.

17.134 oracle/wss_username_token_over_ssl_service_policy

The oracle/wss_username_token_over_ssl_service_policy uses the credentials in the WS-Security UsernameToken SOAP header to authenticate users against the OPSS configured identity store, and verifies that the transport protocol provides SSL message protection.

Display Name: Wss Username Token Over SSL Service Policy

Category: Security

Description

Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based endpoint.

To protect against replay attacks, the assertion provides the option to require nonce and 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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_username_token_over_ssl_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-72](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- The username and password must exist and be valid.
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.

17.135 oracle/ wss_username_token_over_ssl_wssc_client_policy

The oracle/wss_username_token_over_ssl_wssc_client_policy includes credentials in the WS-Security UsernameToken header in outbound SOAP request messages, and verifies that the transport protocol provides SSL message protection.

Display Name: Wss Username Token Over SSL with secure conversation enabled Client Policy

Category: Security

Description

Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

To protect against replay attacks, the assertion provides the option to require nonce and 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 has secure conversation enabled. For more information, see [Configuring Secure Conversation Using Oracle Web Services Manager](#).

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_username_token_over_ssl_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-71](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure secure conversation, as describe in [Configuring Secure Conversation Using Oracle Web Services Manager](#).
- Specify a value for `csf-key`, as described in "[Overriding Policy Configuration Properties](#)". The value signifies a key that maps to a username/password. See "[Adding Keys and User Credentials to Configure the Credential Store](#)" for information about 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.
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Include a WS-Security UsernameToken element (`<wsse:UsernameToken/>`) in the SOAP request message. The client provides a username and password for authentication.

17.136 oracle/ wss_username_token_over_ssl_wssc_service_policy

The oracle/wss_username_token_over_ssl_wssc_service_policy uses the credentials in the WS-Security UsernameToken SOAP header to authenticate users against the OPSS configured identity store, and verifies that the transport protocol provides SSL message protection.

Display Name: Wss Username Token Over SSL with secure conversation enabled Service Policy

Category: Security

Description

Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based endpoint.

To protect against replay attacks, the assertion provides the option to require nonce and 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 has secure conversation enabled. For more information, see [Configuring Secure Conversation Using Oracle Web Services Manager](#).

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_username_token_over_ssl_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-72](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- The username and password must exist and be valid.
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.

17.137 oracle/ wss_username_token_over_ssl_notimestamp_client_policy

Display Name: Wss Username Token Over SSL No Timestamp Client Policy

Category: Security

Description

The oracle/wss_username_token_over_ssl_notimestamp_client_policy includes credentials in the WS-Security UsernameToken header in outbound SOAP request messages. Only plain text mechanism is supported. The credentials can be provided either programmatically through the Java Authentication and Authorization Service (JAAS) subject, or by a reference in the policy to the configured credential store. The policy also verifies that the transport protocol provides SSL message protection. This policy can be attached to any SOAP-based client. Timestamp is not added to the message.

To protect against replay attacks, the assertion provides the option to require nonce and 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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_username_token_over_ssl_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-71](#). For more information, see ["Overriding Policy Configuration Properties"](#).
- If you specify a password type of None on the **Settings** page, you do not need to include a password in the key.
- Specify a value for `csf-key`, as described in ["Overriding Policy Configuration Properties"](#). The value signifies a key that maps to a username/password. For information about how to add the key to the credential store, see ["Adding Keys and User Credentials to Configure the Credential Store"](#).
- Configure one-way or two-way SSL, as described in ["Configuring One-Way SSL on WebLogic Server"](#) or ["Configuring Two-Way SSL for a Web Service Client"](#), respectively.

Design Time Considerations

At design time:

- Override configuration settings, as described in ["About Overriding Client Policy Configuration Properties at Design Time"](#).
- Include a WS-Security UsernameToken element (`<wsse:UsernameToken/>`) in the SOAP request message. The client provides a username and password for authentication.

17.138 oracle/ wss_username_token_over_ssl_notimestamp_service_policy

Display Name: Wss Username Token Over SSL No Timestamp Service Policy

Category: Security

Description

The oracle/wss_username_token_over_ssl_notimestamp_service_policy uses the credentials in the UsernameToken WS-Security SOAP header to authenticate users against the configured identity store. Only plain text mechanism is supported. The policy verifies that the transport protocol provides SSL message protection. This policy can be attached to any SOAP-based endpoint. Timestamp should not be present in the incoming message.

To protect against replay attacks, the assertion provides the option to require nonce and 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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_username_token_over_ssl_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-72](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- The username and password must exist and be valid.
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.

17.139 oracle/ wss10_saml_hok_token_with_message_protection_client_policy

The oracle/wss10_saml_hok_token_with_message_protection_client_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.

Display Name: Wss10 SAML Holder-Of-Key Token With Message Protection Client Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml_hok_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-74](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the policy assertion for message signing, message encryption, or both.
- Specify a value for `saml.issuer.name`, as described in "[Overriding Policy Configuration Properties](#)". The `saml.issuer.name` property defaults to a value of `www.oracle.com`. See "[Adding an Additional SAML Assertion Issuer Name](#)" 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 "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".
- Override the `saml.assertion.filename` property to point to the file that has the holder-of-key assertion, as described in "[Overriding Policy Configuration Properties](#)".
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. 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. For more information, see "[About Overriding Client Policy Configuration Properties at Design Time](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure SAML for the web service client at design time, as described in "[Configuring SAML Web Service Client at Design Time](#)".
- Configure the policy assertion for message signing, message encryption, or both.

The [signature example](#) 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.

The [encryption example](#) shows 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.

17.140 oracle/wss10_saml_hok_token_with_message_protection_service_policy

The oracle/wss10_saml_hok_token_with_message_protection_service_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.

Display Name: Wss10 SAML Holder-Of-Key Token With Message Protection Service Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml_hok_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-75](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml.loginmodule` login module. See "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)" for more information. The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".

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.

- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- To set up OPSS:
 - Configure SAML, as described in "[About SAML Configuration](#)".
 - Configure the policy assertion for message signing, message encryption, or both.
 - 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. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.

- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

17.141 oracle/ wss10_saml_token_with_message_integrity_client_policy

The `oracle/wss10_saml_token_with_message_integrity_client_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.

Display Name: Wss10 SAML Token With Message Integrity Client Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-77](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Specify a value for `saml.issuer.name`, as described in "[Overriding Policy Configuration Properties](#)". The `saml.issuer.name` property defaults to a value of `www.oracle.com`. For additional considerations, see "[Adding an Additional SAML Assertion Issuer Name](#)".
- Specify a value for `user.roles.include`, as described in "[Overriding Policy Configuration Properties](#)".
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. See "[Propagating Identity Context Using SAML Policies](#)" for additional considerations.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)". For more information, see "[Overriding Policy Configuration Properties](#)".
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Configure SAML, as described in "[About SAML Configuration](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. The policy specifically

requires that the client's and web service's respective keystores already contain digital certificates containing each other's public key.

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure the client for SAML at design time, as described in "[Configuring SAML Web Service Client at Design Time](#)".
- 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`.

The [signature example](#) 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.

The [encryption example](#) shows 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.

17.142 oracle/ wss10_saml_token_with_message_integrity_service_policy

The `oracle/wss10_saml_token_with_message_integrity_service_policy` enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

Display Name: Wss10 SAML Token With Message Integrity Service Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-78](#). For more information, see "[Overriding Policy Configuration Properties](#)".

- Override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "[Overview of Policy Configuration Overrides](#)".
- Configure SAML, as described in "[About SAML Configuration](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml.loginmodule` login module. For more information, see "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. See "[Propagating Identity Context Using SAML Policies](#)" for additional considerations.

17.143 oracle/wss10_saml_token_with_message_protection_client_policy

The `oracle/wss10_saml_token_with_message_protection_client_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.

Display Name: Wss10 SAML Token With Message Protection Client Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-77](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the policy assertion for message signing, message encryption, or both.

- Specify a value for `saml.issuer.name`, as described in ["Overriding Policy Configuration Properties"](#). The `saml.issuer.name` property defaults to a value of `www.oracle.com`. For additional considerations, see ["Adding an Additional SAML Assertion Issuer Name"](#).
- Specify a value for `user.roles.include`, as described in ["Overriding Policy Configuration Properties"](#).
- Specify a value for `propagate.identity.context`, as described in ["Overriding Policy Configuration Properties"](#). The `propagate.identity.context` property defaults to a value of blank. See ["Propagating Identity Context Using SAML Policies"](#) for additional considerations.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in ["Overriding Policy Configuration Properties"](#). For more information, see ["Overriding Policy Configuration Properties"](#).
- The web service's base64-encoded public certificate is published in the WSDL for use by the web service client, as described in ["Understanding Service Identity Certificate Extensions"](#). As an alternative, you can specify a value for `keystore.recipient.alias`, as described in ["Overriding Policy Configuration Properties"](#). 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.
- Set up the OWSM keystore, as described in ["Overview of Configuring Keystores for Message Protection"](#).
- Configure SAML, as described in ["About SAML Configuration"](#).
- Set up the web service client keystore, as described in ["Understanding Keys and Certificates"](#) in *Understanding Oracle Web Services Manager*. The policy specifically requires that the client's and web service's respective keystores already contain digital certificates containing each other's public key.

Design Time Considerations

At design time:

- Override configuration settings, as described in ["About Overriding Client Policy Configuration Properties at Design Time"](#).
- Configure the client for SAML at design time, as described in ["Configuring SAML Web Service Client at Design Time"](#).
- Configure the policy assertion for message signing, message encryption, or both.

[The signature example](#) 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.

[The encryption example](#) shows 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.

17.144 oracle/ wss10_saml_token_with_message_protection_service_policy

The oracle/wss10_saml_token_with_message_protection_service_policy enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

Display Name: Wss10 SAML Token With Message Protection Service Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-78](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the policy assertion for message signing, message encryption, or both.
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".

- Configure the `saml.loginmodule` login module. For more information, see "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. The policy specifically requires that the client's and web service's respective keystores already contain digital certificates containing each other's public key.
- To set up OPSS:
 - Configure SAML, as described in "[About SAML Configuration](#)".
 - Configure the policy assertion for message signing, message encryption, or both.
 - 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. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

17.145 oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy

The `oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy` provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

Display Name: Wss10 SAML Token With Message Protection SKI Basic 256 Client Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml_token_with_message_protection_client_template](#)

This assertion is advertised.

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.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-77](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the policy assertion for message signing, message encryption, or both.
- Configure SAML, as described in "[About SAML Configuration](#)".
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- The web service's base64-encoded public certificate is published in the WSDL for use by the web service client, as described in "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `saml.issuer.name`, as described in "[Overriding Policy Configuration Properties](#)". The `saml.issuer.name` property defaults to a value of `www.oracle.com`. For additional considerations, see "[Adding an Additional SAML Assertion Issuer Name](#)".
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

- Specify a value for `user.roles.include`, as described in "[Overriding Policy Configuration Properties](#)".
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. The policy specifically requires that the client's and web service's respective keystores already contain digital certificates containing each other's public key.

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure SAML, as described in "[Configuring SAML Web Service Client at Design Time](#)".
- Configure the policy assertion for message signing, message encryption, or both.

The [signature example](#) 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.

The [encryption example](#) shows 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.

17.146 oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy

The oracle/wss10_saml_token_with_message_protection_ski_basic256_service_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 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.

Display Name: Wss10 SAML Token With Message Protection SKI Basic 256 Service Policy

Category: Security

Description

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 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](#)"

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

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.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-78](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml.loginmodule` login module. For more information, see "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- Override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "[Overview of Policy Configuration Overrides](#)".
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".

- To set up OPSS:
 - Configure SAML, as described in "[About SAML Configuration](#)".
 - 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. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.

17.147 oracle/ wss10_saml20_token_with_message_protection_client_policy

The `oracle/wss10_saml20_token_with_message_protection_client_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 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.

Display Name: Wss10 SAML V2.0 Token With Message Protection Client Policy

Category: Security

Description

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.

For more information about the available algorithms for message protection, see "[Supported Algorithm Suites](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml20_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-80](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure SAML, as described in "[About SAML Configuration](#)".
- Configure the policy assertion for message signing, message encryption, or both.

- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".
- Specify a value for `user.roles.include`, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Specify a value for `propagate.identity.context`, as described in "[Overview of Configuring Keystores for Message Protection](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".
- The web service's base64-encoded public certificate is published in the WSDL for use by the web service client, as described in "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure SAML, as described in "[Configuring SAML Web Service Client at Design Time](#)".
- Set up the web service client keystore, as described in [Understanding Web Service Security Concepts](#). 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 [signature example](#) 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.

The [encryption example](#) shows 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.

17.148 oracle/ wss10_saml20_token_with_message_protection_service_policy

The `oracle/wss10_saml20_token_with_message_protection_service_policy` enforces message protection (integrity and confidentiality) and SAML-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.

Display Name: Wss10 SAML V2.0 Token With Message Protection Service Policy

Category: Security

Description

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.

For more information about the available algorithms for message protection, see "[Supported Algorithm Suites](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_saml20_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-81](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml2.loginmodule` login module. See "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)" for more information. The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- Override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "[Overview of Policy Configuration Overrides](#)".
- To set up OPSS:
 - Configure SAML, as described in "[About SAML Configuration](#)".
 - 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. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.

17.149 oracle/wss10_username_id_propagation_with_msg_protection_client_policy

The oracle/wss10_username_id_propagation_with_msg_protection_client_policy provides message protection (integrity and confidentiality) and identity propagation for outbound SOAP requests in accordance with the WS-Security 1.0 standard. 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.

Display Name: Wss10 Username Id Propagation With Message Protection Client Policy

Category: Security

Note:

In this release, the policy oracle/wss10_username_id_propagation_with_msg_protection_client_policy has been deprecated.

Description

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.

To protect against replay attacks, the assertion provides the option to require nonce and 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.

For more information about the available algorithms for message protection, see "[Supported Algorithm Suites](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_username_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-83](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the policy assertion for message signing, message encryption, or both.
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".

- The web service's base64-encoded public certificate is published in the WSDL for use by the web service client, as described in "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. 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 UsernameToken element (`<wsse:UsernameToken/>`) in the SOAP request message. The client provides a username and password for authentication.

The [signature example](#) 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.

The [encryption example](#) shows 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.

17.150 oracle/wss10_username_id_propagation_with_msg_protection_service_policy

The `oracle/wss10_username_id_propagation_with_msg_protection_service_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. 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.

Display Name: Wss10 Username Id Propagation With Message Protection Service Policy

Category: Security

Note:

In this release, the policy `oracle/wss10_username_id_propagation_with_msg_protection_service_policy` has been deprecated.

Description

This policy can be enforced on any SOAP-based endpoint.

To protect against replay attacks, the assertion provides the option to require nonce and 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.

For more information about the available algorithms for message protection, see "[Supported Algorithm Suites](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_username_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-84](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)". The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
 - Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "[Overview of Policy Configuration Overrides](#)".

17.151 oracle/ wss10_username_token_with_message_protection_client_policy

The `oracle/wss10_username_token_with_message_protection_client_policy` provides message protection (integrity and confidentiality) and authentication 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 mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption.

Display Name: Wss10 Username Token With Message Protection Client Policy

Category: Security

Description

Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

To protect against replay attacks, the assertion provides the option to require nonce and 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.

For more information about the available algorithms for message protection, see "[Supported Algorithm Suites](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_username_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-83](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the policy assertion for message signing, message encryption, or both.
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- The web service's base64-encoded public certificate is published in the WSDL for use by the web service client, as described in "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `csf-key`, as described in "[Overriding Policy Configuration Properties](#)". The value signifies a key that maps to a username/password. For more information about the how to add the key to the credential store, see "[Adding Keys and User Credentials to Configure the Credential Store](#)".
- Override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "[Overview of Policy Configuration Overrides](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Set up the web service client keystore, as described in [Understanding Web Service Security Concepts](#). 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.

The [signature example](#) 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.

The [encryption example](#) shows 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.

17.152 oracle/wss10_username_token_with_message_protection_service_policy

The oracle/wss10_username_token_with_message_protection_service_policy enforces message protection (message integrity and confidentiality) and 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 mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption.

Display Name: Wss10 Username Token With Message Protection Service Policy

Category: Security

Description

Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based endpoint.

To protect against replay attacks, the assertion provides the option to require nonce and 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.

For more information about the available algorithms for message protection, see "[Supported Algorithm Suites](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_username_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-84](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.

- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
- Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
- Override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "[Overview of Policy Configuration Overrides](#)".

17.153 oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy

The `oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy` provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. 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.

Display Name: Wss10 Username Token With Message Protection SKI Basic 256 Client Policy

Category: Security

Description

Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

To protect against replay attacks, the assertion provides the option to require nonce and 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 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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_username_token_with_message_protection_client_template](#)

This assertion is advertised.

 **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.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-83](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the policy assertion for message signing, message encryption, or both.
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- The web service's base64-encoded public certificate is published in the WSDL for use by the web service client, as described in "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `csf-key`, as described in "[Overriding Policy Configuration Properties](#)". The value signifies a key that maps to a username/password. For more information about the how to add the key to the credential store, see "[Adding Keys and User Credentials to Configure the Credential Store](#)".
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. 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.

The [signature example](#) 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.

The [encryption example](#) 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.

17.154 oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy

The oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy enforces message protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. 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.

Display Name: Wss10 Username Token With Message Protection SKI Basic 256 Service Policy

Category: Security

Description

Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based endpoint.

To protect against replay attacks, the assertion provides the option to require nonce and 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 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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_username_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

 **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.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-84](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - 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. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "[Overview of Policy Configuration Overrides](#)".

17.155 oracle/ wss10_x509_token_with_message_protection_client_policy

The oracle/wss10_x509_token_with_message_protection_client_policy provides message protection (integrity and confidentiality) and certificate credential population for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

Display Name: Wss10 X509 Token With Message Protection Client Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_x509_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-86](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the policy assertion for message signing, message encryption, or both.
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- The web service's base64-encoded public certificate is published in the WSDL for use by the web service client, as described in "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. The policy specifically

requires that the client's and web service's respective keystores already contain digital certificates containing each other's public key.

- Provide valid X.509 authentication credentials in the SOAP message through the WS-Security binary security token.
- Configure the policy assertion for message signing, message encryption, or both.

The [signature example](#) 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.

The [encryption example](#) shows 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.

17.156 oracle/wss10_x509_token_with_message_protection_service_policy

The oracle/wss10_x509_token_with_message_protection_service_policy enforces message protection (integrity and confidentiality) and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

Display Name: Wss10 X509 Token With Message Protection Service Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss10_x509_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-87](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure an Authentication provider, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".

- Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
- Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
- Override the `keystore.sig.csf.key` and `keystore.enc.csf.key` server-side configuration properties, as described in "[Overview of Policy Configuration Overrides](#)".

17.157 oracle/ wss11_kerberos_token_with_message_protection_client_policy

The `oracle/wss11_kerberos_token_with_message_protection_client_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.

Display Name: Wss11 Kerberos Token With Message Protection Client Policy

Category: Security

Description

This policy can be enforced on any SOAP-based client.

This policy is compatible with MIT Kerberos KDC and with newer versions of Active Directory KDC. It is not compatible with versions of Active Directory earlier than 2008 because it uses Triple DES encryption. With these earlier versions, use "[oracle/wss11_kerberos_token_with_message_protection_basic128_client_policy](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_kerberos_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-89](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the policy assertion for message signing, message encryption, or both.
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".
- Configure Kerberos tokens, as described in "[Understanding Kerberos Token Configuration](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. 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.

The [encryption example](#) shows 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.

17.158 oracle/ wss11_kerberos_token_with_message_protection_service_policy

The oracle/wss11_kerberos_token_with_message_protection_service_policy is enforced 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, and it enforces message integrity and confidentiality using Kerberos keys. The container must have the Kerberos infrastructure configured through Oracle Platform Security Services.

Display Name: Wss11 Kerberos Token With Message Protection Service Policy

Category: Security

Description

This policy can be enforced on any SOAP-based endpoint.

This policy is compatible with MIT Kerberos KDC and with newer versions of Active Directory KDC. It is not compatible with versions of Active Directory earlier than 2008 because it uses Triple DES encryption. With these earlier versions, use "[oracle/wss11_kerberos_token_with_message_protection_basic128_service_policy](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_kerberos_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-89](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the `krb5.loginmodule` login module. See "[Configuring the Kerberos Login Module](#)".

- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
 - Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Override the `keystore.enc.csf.key` server-side configuration property, as described in "[Overview of Policy Configuration Overrides](#)".
 - Configure Kerberos, as described in "[Understanding Kerberos Token Configuration](#)".

17.159 oracle/wss11_kerberos_token_with_message_protection_basic128_client_policy

The oracle/wss11_kerberos_token_with_message_protection_basic128_client_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.

Display Name: Wss11 Kerberos Token With Message Protection Basic 128 Client Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_kerberos_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-92](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the policy assertion for message signing, message encryption, or both.
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".
- Configure Kerberos tokens, as described in "[Understanding Kerberos Token Configuration](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. 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.

The [encryption example](#) shows 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.

17.160 oracle/wss11_kerberos_token_with_message_protection_basic128_service_policy

The `oracle/wss11_kerberos_token_with_message_protection_basic128_service_policy` is enforced in accordance with the WS-Security Kerberos Token Profile v1.1 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.

Display Name: Wss11 Kerberos Token With Message Protection Basic 128 Service Policy

Category: Security

Description

This policy is compatible with Active Directory KDCs. This policy can be attached to any SOAP-based endpoint.

For more information about the available algorithms for message protection, see "[Supported Algorithm Suites](#)".

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 an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_kerberos_token_with_message_protection_service_template](#)

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-93](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the `krb5.loginmodule` login module. See "[Configuring the Kerberos Login Module](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
 - Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Configure Kerberos, as described in "[Understanding Kerberos Token Configuration](#)".
 - Override the `keystore.enc.csf.key` server-side configuration property, as described in "[Overview of Policy Configuration Overrides](#)".

17.161 oracle/wss11_saml_or_username_token_with_message_protection_service_policy

Display Name: Wss11 SAML Token or Wss11 Username Token With Message Protection or Wss SAML Token (Confirmation Method As Bearer) Over SSL or Wss Username Token Over SSL or Http Basic Auth Over SSL or HTTP JWT Token Over SSL Service Policy

Category: Security

Description

The `oracle/wss11_saml_or_username_token_with_message_protection_service_policy` enforces message protection (integrity and confidentiality) and an authentication policy, based on whether the client uses a SAML, username, or HTTP token.

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.
- HTTP authentication using the username provided in the JWT token in the HTTP header to authenticates users. This policy also verifies that the transport protocol is HTTPS.

To protect against replay attacks, the assertion provides the option to require nonce and 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 the symmetric key technology for signing and encryption, the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures, the RSA key mechanisms for message confidentiality, the SHA-1 or SHA-2 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "[Supported Algorithm Suites](#)".

Assertions (OR Group)

This policy contains the following assertions, as an OR group—meaning any one of the tokens can be sent by the client:

- [oracle/wss11_saml_token_with_message_protection_service_template](#)
- [oracle/wss11_username_token_with_message_protection_service_template](#)
- [oracle/wss_saml_token_bearer_over_ssl_service_template](#)
- [oracle/wss_username_token_over_ssl_service_template](#)
- [oracle/wss_http_token_over_ssl_service_template](#)
- [oracle/http_jwt_token_over_ssl_service_template](#)

The assertions are advertised in the WSDL.

17.162 oracle/ wss11_saml_or_username_token_with_message_protection_sha 256_service_policy

Display Name: Wss11 Saml Token or Wss11 Username Token With Message Protection or Wss SAML Token (Confirmation Method As Bearer) Over SSL or Wss Username Token Over SSL or Http Basic Auth Over SSL Sha256 or HTTP JWT Token Over SSL Service Policy

Category: Security

Description

The oracle/wss11_saml_or_username_token_with_message_protection_sha256_service_policy enforces message protection (integrity and confidentiality) and an authentication policy, based on whether the client uses a SAML, username, or HTTP token.

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.
- HTTP authentication using the username provided in the JWT token in the HTTP header to authenticates users. This policy also verifies that the transport protocol is HTTPS.

To protect against replay attacks, the assertion provides the option to require nonce and 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 the symmetric key technology for signing and encryption, the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures, specifically RSA key mechanisms for message confidentiality, SHA-2 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "[Supported Algorithm Suites](#)".

Assertions (OR Group)

This policy contains the following assertions, as an OR group—meaning any one of the tokens can be sent by the client:

- [oracle/wss11_saml_token_with_message_protection_service_template](#)
- [oracle/wss11_username_token_with_message_protection_service_template](#)
- [oracle/wss_saml_token_bearer_over_ssl_service_template](#)
- [oracle/wss_username_token_over_ssl_service_template](#)
- [oracle/wss_http_token_over_ssl_service_template](#)
- [oracle/http_jwt_token_over_ssl_service_template](#)

The assertions are advertised in the WSDL.

17.163 oracle/ wss11_saml_token_identity_switch_with_message_protection_client_policy

The oracle/wss11_saml_token_identity_switch_with_message_protection_client_policy enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

Display Name: Wss11 Saml Token Identity Switch With Message Protection Client Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_saml_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-95](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- `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 `csf-key` username property.) 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 OWSM accepts the externally-supplied identity must have the `WSIdentityPermission` permission. This is to avoid potentially rogue applications from providing an identity to OWSM.
- For information about configuring this policy, see "[About SAML Web Service Client Configuration for Identity Switching](#)". In particular, you need to set the `javax.xml.ws.security.auth.username` property, as described in "[Setting the javax.xml.ws.security.auth.username Property](#)", and the `WSIdentityPermission` permission, as described in "[Setting the Permission Using WSIdentityPermission](#)".
- For additional SAML considerations, see "[Configuring SAML Web Service Client at Design Time](#)".

- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- 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 "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".
- Specify a value for `saml.issuer.name`, as described in "[Overriding Policy Configuration Properties](#)". The `saml.issuer.name` property defaults to a value of `www.oracle.com`. For additional considerations, see "[Adding an Additional SAML Assertion Issuer Name](#)".
- Specify a value for `saml.issuer.uri`, as described in "[Overriding Policy Configuration Properties](#)".
- Specify a value for `user.roles.include`, as described in "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. 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 SAML, as described in "[Configuring SAML Web Service Client at Design Time](#)".
- Configure the policy assertion for message signing, message encryption, or both.

The [encryption example](#) shows 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.

17.164 oracle/wss11_saml_token_identity_switch_with_message_protection_sha256_client_policy

Display Name: Wss11 Saml Token Identity Switch With Message Protection Sha256 Client Policy

Category: Security

Description

The oracle/wss11_saml_token_identity_switch_with_message_protection_sha256_client_policy enables

message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

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, specifically RSA key mechanisms for message confidentiality, SHA-2 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. For more information about the available asymmetric algorithms for message protection, see "[Supported Algorithm Suites](#)".

A SAML token is included in the SOAP message for use in SAML based authentication with sender vouches confirmation. These credentials are provided either programmatically or through the security configuration. This policy performs dynamic identity switching by propagating a different identity than the one based on authenticated Subject. This policy can be attached to any SOAP-based client.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_saml_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-95](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- `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 `csf-key` username property.) 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 OWSM accepts the externally-supplied identity must have the `WSIdentityPermission` permission. This is to avoid potentially rogue applications from providing an identity to OWSM.
- For information about configuring this policy, see "[About SAML Web Service Client Configuration for Identity Switching](#)". In particular, you need to set the `javax.xml.ws.security.auth.username` property, as described in "[Setting the javax.xml.ws.security.auth.username Property](#)", and the `WSIdentityPermission` permission, as described in "[Setting the Permission Using WSIdentityPermission](#)".
- For additional SAML considerations, see "[Configuring SAML Web Service Client at Design Time](#)".
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- 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 "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.

- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".
- Specify a value for `saml.issuer.name`, as described in "[Overriding Policy Configuration Properties](#)". The `saml.issuer.name` property defaults to a value of `www.oracle.com`. For additional considerations, see "[Adding an Additional SAML Assertion Issuer Name](#)".
- Specify a value for `saml.issuer.uri`, as described in "[Overriding Policy Configuration Properties](#)".
- Specify a value for `user.roles.include`, as described in "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. 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 SAML, as described in "[Configuring SAML Web Service Client at Design Time](#)".
- Configure the policy assertion for message signing, message encryption, or both.

The [encryption example](#) shows 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.

17.165 oracle/ wss11_saml_token_with_message_protection_client_policy

The `oracle/wss11_saml_token_with_message_protection_client_policy` enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

Display Name: Wss11 Saml Token With Message Protection Client Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_saml_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-95](#). For more information, see ["Overriding Policy Configuration Properties"](#).
- Set up the OWSM keystore, as described in ["Overview of Configuring Keystores for Message Protection"](#).
- 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 ["Understanding Service Identity Certificate Extensions"](#). As an alternative, you can specify a value for `keystore.recipient.alias`, as described in ["Overriding Policy Configuration Properties"](#). 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.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in ["Overriding Policy Configuration Properties"](#).
- Specify a value for `saml.issuer.name`, as described in ["Overriding Policy Configuration Properties"](#). The `saml.issuer.name` property defaults to a value of `www.oracle.com`. For additional considerations, see ["Adding an Additional SAML Assertion Issuer Name"](#).
- Specify a value for `user.roles.include`, as described in ["Overriding Policy Configuration Properties"](#).
- Specify a value for `propagate.identity.context`, as described in ["Overview of Configuring Keystores for Message Protection"](#). The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see ["Propagating Identity Context Using SAML Policies"](#).

Design Time Considerations

At design time:

- Override configuration settings, as described in ["About Overriding Client Policy Configuration Properties at Design Time"](#).
- Set up the web service client keystore, as described in ["Understanding Keys and Certificates"](#) in *Understanding Oracle Web Services Manager*. 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 SAML, as described in ["Configuring SAML Web Service Client at Design Time"](#).
- Configure the policy assertion for message signing, message encryption, or both.

[The encryption example](#) shows 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.

17.166 oracle/ wss11_saml_token_with_message_protection_service_policy

The oracle/wss11_saml_token_with_message_protection_service_policy enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

Display Name: Wss11 SAML Token With Message Protection Service Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_saml_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-96](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure SAML, as described in "[About SAML Configuration](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
 - Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.

- Store the password for the decryption key in the credential store, as described in ["Adding Keys and User Credentials to Configure the Credential Store"](#). Use `keystore.enc.csf.key` as the key name.
- Override the `keystore.enc.csf.key` server-side configuration property, as described in ["Overview of Policy Configuration Overrides"](#).

17.167 oracle/wss11_saml_token_with_message_protection_sha256_client_policy

Display Name: Wss11 Saml Token With Message Protection Sha256 Client Policy

Category: Security

Description

The `oracle/wss11_saml_token_with_message_protection_sha256_client_policy` enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of symmetric key technology for endorsing signatures, the RSA key mechanisms for message confidentiality, the SHA-2 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available asymmetric algorithms for message protection, see ["Supported Algorithm Suites"](#).

The keystore on the client is configured either on a per-request basis or through the security configuration. A SAML token is included in the SOAP message for use in SAML based authentication with sender vouches confirmation. These credentials are provided either programmatically or through the security configuration. This policy can be attached to any SOAP-based client.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_saml_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-95](#). For more information, see ["Overriding Policy Configuration Properties"](#).
- Set up the OWSM keystore, as described in ["Overview of Configuring Keystores for Message Protection"](#).
- 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 ["Understanding Service Identity Certificate Extensions"](#). As an alternative, you can specify a value for `keystore.recipient.alias`, as described in ["Overriding Policy Configuration Properties"](#). 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.

- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in ["Overriding Policy Configuration Properties"](#).
- Specify a value for `saml.issuer.name`, as described in ["Overriding Policy Configuration Properties"](#). The `saml.issuer.name` property defaults to a value of `www.oracle.com`. For additional considerations, see ["Adding an Additional SAML Assertion Issuer Name"](#).
- Specify a value for `user.roles.include`, as described in ["Overriding Policy Configuration Properties"](#).
- Specify a value for `propagate.identity.context`, as described in ["Overview of Configuring Keystores for Message Protection"](#). The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see ["Propagating Identity Context Using SAML Policies"](#).

Design Time Considerations

At design time:

- Override configuration settings, as described in ["About Overriding Client Policy Configuration Properties at Design Time"](#).
- Set up the web service client keystore, as described in ["Understanding Keys and Certificates"](#) in *Understanding Oracle Web Services Manager*. 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 SAML, as described in ["Configuring SAML Web Service Client at Design Time"](#).
- Configure the policy assertion for message signing, message encryption, or both.

The [encryption example](#) shows 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.

17.168 oracle/wss11_saml_token_with_message_protection_sha256_service_policy

Display Name: Wss11 Saml Token With Message Protection Sha256 Service Policy

Category: Security

Description

The `oracle/wss11_saml_token_with_message_protection_sha256_service_policy` enables message protection (integrity and confidentiality) and SAML token population for inbound SOAP requests using mechanisms described in WS-Security 1.1.

This policy uses the symmetric key technology for signing and encryption, the WS-Security's Basic 128 suite of symmetric key technology for endorsing signatures, the RSA key mechanisms for message confidentiality, the SHA-2 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available asymmetric algorithms for message protection, see ["Supported Algorithm Suites"](#).

The keystore is configured through the security configuration. It extracts the SAML token from the WS-Security binary security token, and uses those credentials to validate users against the configured identity store. This policy can be attached to any SOAP-based endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_saml_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-96](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure SAML, as described in "[About SAML Configuration](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
 - Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Override the `keystore.enc.csf.key` server-side configuration property, as described in "[Overview of Policy Configuration Overrides](#)".

17.169 oracle/wss11_saml_token_with_message_protection_wssc_client_policy

The `oracle/wss11_saml_token_with_message_protection_wssc_client_policy` enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

Display Name: Wss11 Saml Token With Message Protection with secure conversation enabled Client Policy

Category: Security

Description

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](#)".

This policy has secure conversation enabled.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_saml_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-95](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure Secure Conversation, as described in [Configuring Secure Conversation Using Oracle Web Services Manager](#).
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- 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 "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".
- Specify a value for `saml.issuer.name`, as described in "[Overriding Policy Configuration Properties](#)". The `saml.issuer.name` property defaults to a value of `www.oracle.com`. For additional considerations, see "[Adding an Additional SAML Assertion Issuer Name](#)".
- Specify a value for `user.roles.include`, as described in "[Overriding Policy Configuration Properties](#)".
- Specify a value for `propagate.identity.context`, as described in "[Overview of Configuring Keystores for Message Protection](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. 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 SAML, as described in "[Configuring SAML Web Service Client at Design Time](#)".
- Configure Secure Conversation, as described in [Configuring Secure Conversation Using Oracle Web Services Manager](#) .
- Configure the policy assertion for message signing, message encryption, or both.

The [encryption example](#) shows 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.

17.170 oracle/wss11_saml_token_with_message_protection_wssc_service_policy

The oracle/wss11_saml_token_with_message_protection_wssc_service_policy enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

Display Name: Wss11 Saml Token With Message Protection with secure conversation enabled Service Policy

Category: Security

Description

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](#)".

This policy has secure conversation enabled. See [Configuring Secure Conversation Using Oracle Web Services Manager](#) .

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_saml_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-96](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure SAML, as described in "[About SAML Configuration](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
 - Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Override the `keystore.enc.csf.key` server-side configuration property, as described in "[Overview of Policy Configuration Overrides](#)".

17.171 oracle/wss11_saml_token_with_message_protection_wssc_reauthn_client_policy

The `oracle/wss11_saml_token_with_message_protection_wssc_reauthn_client_policy` enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

Display Name: Wss11 Saml Token With Message Protection with secure conversation and re-authenticate mode enabled Client Policy

Category: Security

Description

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](#)".

This policy has secure conversation enabled.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_saml_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-95](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure Secure Conversation, as described in [Configuring Secure Conversation Using Oracle Web Services Manager](#).
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- 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 "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".
- Specify a value for `saml.issuer.name`, as described in "[Overriding Policy Configuration Properties](#)". The `saml.issuer.name` property defaults to a value of `www.oracle.com`. For additional considerations, see "[Adding an Additional SAML Assertion Issuer Name](#)".
- Specify a value for `user.roles.include`, as described in "[Overriding Policy Configuration Properties](#)".
- Specify a value for `propagate.identity.context`, as described in "[Overview of Configuring Keystores for Message Protection](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Set up the web service client keystore, as described in "Understanding Keys and Certificates" in *Understanding Oracle Web Services Manager*. 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 SAML, as described in "[Configuring SAML Web Service Client at Design Time](#)".
- Configure Secure Conversation, as described in [Configuring Secure Conversation Using Oracle Web Services Manager](#).

- Configure the policy assertion for message signing, message encryption, or both.

The [encryption example](#) 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.

17.172 oracle/wss11_saml_token_with_message_protection_wssc_reauthn_service_policy

The oracle/wss11_saml_token_with_message_protection_wssc_reauthn_service_policy enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

Display Name: Wss11 Saml Token With Message Protection with secure conversation and re-authenticate mode enabled Service Policy

Category: Security

Description

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](#)".

This policy has secure conversation enabled. See [Configuring Secure Conversation Using Oracle Web Services Manager](#) .

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_saml_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-96](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure SAML, as described in "[About SAML Configuration](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- To set up OPSS:

- Configure the policy assertion for message signing, message encryption, or both.
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
- Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
- Override the `keystore.enc.csf.key` server-side configuration property, as described in "[Overview of Policy Configuration Overrides](#)".

17.173 oracle/ wss11_saml20_token_with_message_protection_client_policy

The `oracle/wss11_saml20_token_with_message_protection_client_policy` enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

Display Name: Wss11 Saml V2.0 Token With Message Protection Client Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_saml20_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-98](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- 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 "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as

described in ["Overriding Policy Configuration Properties"](#). 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.

- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in ["Overriding Policy Configuration Properties"](#).
- Specify a value for `saml.issuer.name`, as described in ["Overriding Policy Configuration Properties"](#). The `saml.issuer.name` property defaults to a value of `www.oracle.com`. For additional considerations, see ["Adding an Additional SAML Assertion Issuer Name"](#).
- Specify a value for `user.roles.include`, as described in ["Overriding Policy Configuration Properties"](#).
- Specify a value for `propagate.identity.context`, as described in ["Overview of Configuring Keystores for Message Protection"](#). The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see ["Propagating Identity Context Using SAML Policies"](#).

Design Time Considerations

At design time:

- Override configuration settings, as described in ["About Overriding Client Policy Configuration Properties at Design Time"](#).
- Set up the web service client keystore, as described in ["Understanding Keys and Certificates"](#) in *Understanding Oracle Web Services Manager*. 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 SAML, as described in ["Configuring SAML Web Service Client at Design Time"](#).
- Configure Secure Conversation, as described in ["Configuring Secure Conversation Using Oracle Web Services Manager"](#).
- Configure the policy assertion for message signing, message encryption, or both.

The [encryption example](#) shows 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.

17.174 oracle/ wss11_saml20_token_with_message_protection_service_policy

The `oracle/wss11_saml20_token_with_message_protection_service_policy` enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

Display Name: Wss11 Saml V2.0 Token With Message Protection Service Policy

Category: Security

Description

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](#)".

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_saml20_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-99](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure SAML, as described in "[About SAML Configuration](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
 - Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Override the `keystore.enc.csf.key` server-side configuration property, as described in "[Overview of Policy Configuration Overrides](#)".

17.175 oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy

The Wss11 Issued Token with Saml Holder of Key with Message Protection Client Policy inserts a SAML HOK assertion issued by a trusted STS (Security Token Service).

Display Name: Wss11 Issued Token with Saml Holder of Key with Message Protection Client Policy

Category: Security

Description

Inserts a SAML HOK assertion issued by a trusted STS (Security Token Service). Messages are protected using proof key material provided by the STS.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-118](#). For more information, see ["Overriding Policy Configuration Properties"](#).

Note:

When using Oracle STS, the asymmetric proof key (HoK) use-case works only when a client cert csf key is configured in the policy using the `sts.auth.x509.csf.key` configuration override.

This value is used for signing the WS-Trust request sent to the STS and by Oracle STS as the proof key. The public key in the SAML assertion also corresponds to this keypair.

- Set up the web service client, as described in ["Main Steps in Setting Up Automatic Policy Configuration"](#).
- Set up the OWSM keystore to specify a key (username/password or X.509) to authenticate to the STS, as described in ["Overview of Configuring Keystores for Message Protection"](#).
- 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 ["Understanding Service Identity Certificate Extensions"](#). As an alternative, you can specify a value for `keystore.recipient.alias`, as described in ["Overriding Policy Configuration Properties"](#). 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.

- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "Overriding Policy Configuration Properties".

Design Time Considerations

At design time:

- Override configuration settings, as described in "About Overriding Client Policy Configuration Properties at Design Time". For examples of overriding STS configuration settings, see "Programmatically Overriding Policy Configuration for WS-Trust Client Policies".
- Set up the web service client, as described in "Main Steps in Setting Up Automatic Policy Configuration".
- Set up the OWSM keystore to specify a key (username/password or X.509) to authenticate to the STS, as described in "Overview of Configuring Keystores for Message Protection".
- Configure the policy assertion for message signing, message encryption, or both.

17.176 oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy

The `oracle/wss11_sts_issued_saml_hok_with_message_protection_service_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.

Display Name: Wss11 Issued Token with Saml Holder of Key with Message Protection Service Policy

Category: Security

Description

You also have the option to override the `keystore.enc.csf.key` server-side configuration property, as described in "Overview of Policy Configuration Overrides".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-119](#). For more information, see "Overriding Policy Configuration Properties".
- Set up the web service, as described in "Main Steps in Setting Up Automatic Policy Configuration".

- Specify a value for `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

17.177 oracle/wss11_sts_issued_saml_hok_with_message_protection_client_policy

The Wss11 Issued Token with Saml Holder of Key with Message Protection Client Policy inserts a SAML HOK assertion issued by a trusted STS (Security Token Service).

Display Name: Wss11 Issued Token with Saml Holder of Key with Message Protection Client Policy

Category: Security

Description

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-118](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Set up the web service client, as described in "[Main Steps in Setting Up Automatic Policy Configuration](#)".
- Set up the OWSM keystore to specify a key (username/password or X.509) to authenticate to the STS, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- 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 "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)". For examples of overriding STS configuration settings, see "[Programmatically Overriding Policy Configuration for WS-Trust Client Policies](#)".
- Set up the web service client, as described in "[Main Steps in Setting Up Automatic Policy Configuration](#)".
- Set up the OWSM keystore to specify a key (username/password or X.509) to authenticate to the STS, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Configure the policy assertion for message signing, message encryption, or both.

17.178 oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy

The oracle/wss11_sts_issued_saml_hok_with_message_protection_service_policy authenticates a SAML HOK assertion issued by a trusted STS (Security Token Service).

Display Name: Wss11 Issued Token with Saml Holder of Key with Message Protection Service Policy

Category: Security

Description

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-119](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Set up the web service, as described in "[Main Steps in Setting Up Automatic Policy Configuration](#)".
- Specify a value for `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

17.179 oracle/ wss11_sts_issued_saml_with_message_protection_client_policy

The Wss11 Issued Token with Saml Sender Vouches with Message Protection Client Policy inserts a SAML sender vouches assertion issued by a trusted STS (Security Token Service).

Display Name: Wss11 Issued Token with Saml Sender Vouches with Message Protection Client Policy

Category: Security

Description

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.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_sts_issued_saml_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-121](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Set up the web service client, as described in "[Main Steps in Setting Up Automatic Policy Configuration](#)".
- Set up the OWSM keystore to specify a key (username/password or X.509) to authenticate to the STS, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- 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 "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)". For examples of overriding STS configuration settings, see "[Programmatically Overriding Policy Configuration for WS-Trust Client Policies](#)".

- Set up the web service client, as described in "[Main Steps in Setting Up Automatic Policy Configuration](#)".
- Set up the OWSM keystore to specify a key (username/password or X.509) to authenticate to the STS, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Configure the policy assertion for message signing, message encryption, or both.

17.180 oracle/wss11_username_token_with_message_protection_client_policy

The oracle/wss11_username_token_with_message_protection_client_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.

Display Name: Wss11 Username Token With Message Protection Client Policy

Category: Security

Description

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 protect against replay attacks, the assertion provides the option to require nonce and 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.

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_username_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-101](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Set up the OWSM keystore to specify a key (username/password or X.509) to authenticate to the STS, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- 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 "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties 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.

The [encryption example](#) shows 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.

17.181 oracle/wss11_username_token_with_message_protection_service_policy

The `oracle/wss11_username_token_with_message_protection_service_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.

Display Name: Wss11 Username Token With Message Protection Service Policy

Category: Security

Description

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 protect against replay attacks, the assertion provides the option to require nonce and 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.

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_username_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-102](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
 - Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Override the `keystore.enc.csf.key` server-side configuration property, as described in "[Overview of Policy Configuration Overrides](#)".

17.182 oracle/ wss11_username_token_with_message_protection_sha256_client_policy

Display Name: Wss11 Username Token With Message Protection Sha256 Client Policy

Category: Security

Description

The `oracle/wss11_username_token_with_message_protection_sha256_client_policy` provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard. Only plain text mechanism is supported. This policy can be attached to any SOAP-based client.

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 protect against replay attacks, the assertion provides the option to require nonce and 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.

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, specifically RSA key mechanisms for message confidentiality, SHA-2 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available asymmetric algorithms for message protection, see "[Supported Algorithm Suites](#)".

The keystore on the client side is configured either on a per-request basis or through the security configuration. Credentials are included in the WS-Security UsernameToken header of outbound SOAP request messages. Credentials are provided either programmatically through the current Java Authentication and Authorization Service (JAAS) subject or by a reference in the policy to the configured credential store.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_username_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-101](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Set up the OWSM keystore to specify a key (username/password or X.509) to authenticate to the STS, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- 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 "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `keystore.sig.csf.key` and `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties 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.

The [encryption example](#) shows 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.

17.183 oracle/wss11_username_token_with_message_protection_sha256_service_policy

Display Name: Wss11 Username Token With Message Protection Sha256 Service Policy

Category: Security

Description

The oracle/wss11_username_token_with_message_protection_sha256_service_policy enforces message protection (integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard. Only plain text mechanism is supported.

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 protect against replay attacks, the assertion provides the option to require nonce and 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.

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, specifically RSA key mechanisms for message confidentiality, SHA-2 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available asymmetric algorithms for message protection, see "[Supported Algorithm Suites](#)".

The keystore is configured through the security configuration. Credentials are provided through the UsernameToken WS-Security SOAP header. The credentials are authenticated against the configured identity store.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_username_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-102](#). For more information, see "[Overriding Policy Configuration Properties](#)".

- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
 - Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Override the `keystore.enc.csf.key` server-side configuration property, as described in "[Overview of Policy Configuration Overrides](#)".

17.184 oracle/ wss11_username_token_with_message_protection_wssc_client_ policy

The oracle/wss11_username_token_with_message_protection_wssc_client_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.

Display Name: Wss11 Username Token With Message Protection with secure conversation enabled Client Policy

Category: Security

Description

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 protect against replay attacks, the assertion provides the option to require nonce and 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.

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](#)".

This policy has secure conversation enabled.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_username_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-101](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Set up the OWSM keystore to specify a key (username/password or X.509) to authenticate to the STS, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Configure the policy assertion for message signing, message encryption, or both.
- Configure Secure Conversation, as described in [Configuring Secure Conversation Using Oracle Web Services Manager](#).
- The web service's base64-encoded public certificate is published in the WSDL for use by the web service client, as described in "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.
- Specify a value for `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties 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 Secure Conversation, as described in [Configuring Secure Conversation Using Oracle Web Services Manager](#).
- Configure the policy assertion for message signing, message encryption, or both.

[The encryption example](#) shows 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.

17.185 oracle/wss11_username_token_with_message_protection_wssc_service_policy

The oracle/wss11_username_token_with_message_protection_wssc_service_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.

Display Name: Wss11 Username Token With Message Protection with secure conversation enabled Service Policy

Category: Security

Description

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 protect against replay attacks, the assertion provides the option to require nonce and 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.

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](#)".

This policy has secure conversation enabled. See [Configuring Secure Conversation Using Oracle Web Services Manager](#) .

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_username_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-102](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- To set up OPSS:

- Configure the policy assertion for message signing, message encryption, or both.
- Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
- Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
- Override the `keystore.enc.csf.key` server-side configuration property, as described in "[Overview of Policy Configuration Overrides](#)".

17.186 oracle/ wss11_x509_token_with_message_protection_client_policy

The `oracle/wss11_x509_token_with_message_protection_client_policy` provides message protection (integrity and confidentiality) and certificate-based authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

Display Name: Wss11 X509 Token With Message Protection Client Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_x509_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-104](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Set up the OWSM keystore to specify a key (username/password or X.509) to authenticate to the STS, as described in "[Overview of Configuring Keystores for Message Protection](#)".
- 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 "[Understanding Service Identity Certificate Extensions](#)". As an alternative, you can specify a value for `keystore.recipient.alias`, as described in "[Overriding Policy Configuration Properties](#)". 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.

- Specify a value for `keystore.enc.csf.key`, as described in "[Overriding Policy Configuration Properties](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Set up the web service client keystore, as described in Understanding Web Service Security Concepts. 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.
- Configure the policy assertion for message signing, message encryption, or both.

The [encryption example](#) shows 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.

17.187 oracle/wss11_x509_token_with_message_protection_service_policy

The Wss11 X509 Token With Message Protection Service Policy enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

Display Name: Wss11 X509 Token With Message Protection Service Policy

Category: Security

Description

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](#)".

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_x509_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-105](#). For more information, see "[Overriding Policy Configuration Properties](#)".

- Configure the Authentication provider, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
 - Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Override the `keystore.enc.csf.key` server-side configuration property, as described in "[Overview of Policy Configuration Overrides](#)".

17.188 oracle/wss11_x509_token_with_message_protection_wssc_client_policy

The Wss11 X509 Token With Message Protection with secure conversation enabled Client Policy provides message protection (integrity and confidentiality) and certificate-based authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

Display Name: Wss11 X509 Token With Message Protection with secure conversation enabled Client Policy

Category: Security

Description

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](#)".

This policy has secure conversation enabled.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_x509_token_with_message_protection_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-104](#). For more information, see "[Overriding Policy Configuration Properties](#)".

- Set up the OWSM keystore to specify a key (username/password or X.509) to authenticate to the STS, as described in ["Overview of Configuring Keystores for Message Protection"](#).
- Configure the policy assertion for message signing, message encryption, or both.
- Configure Secure Conversation, as described in [Configuring Secure Conversation Using Oracle Web Services Manager](#).
- The web service's base64-encoded public certificate is published in the WSDL for use by the web service client, as described in ["Understanding Service Identity Certificate Extensions"](#). As an alternative, you can specify a value for `keystore.recipient.alias`, as described in ["Overriding Policy Configuration Properties"](#). 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.

Design Time Considerations

At design time:

- Override configuration settings, as described in ["About Overriding Client Policy Configuration Properties at Design Time"](#).
- Configure Secure Conversation, as described in [Configuring Secure Conversation Using Oracle Web Services Manager](#).
- The web service client needs to provide valid X.509 authentication credentials in the SOAP message through the WS-Security binary security token.
- Set up the web service client keystore, as described in [Understanding Web Service Security Concepts](#). 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.
- Configure the policy assertion for message signing, message encryption, or both.

[The encryption example](#) shows 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.

17.189 oracle/wss11_x509_token_with_message_protection_wssc_service_policy

The Wss11 X509 Token With Message Protection with secure conversation enabled Service Policy enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

Display Name: Wss11 X509 Token With Message Protection with secure conversation enabled Service Policy

Category: Security

Description

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](#)".

This policy has secure conversation enabled. See [Configuring Secure Conversation Using Oracle Web Services Manager](#).

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss11_x509_token_with_message_protection_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-105](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure the Authentication provider, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- To set up OPSS:
 - Configure the policy assertion for message signing, message encryption, or both.
 - Set up the OWSM keystore, as described in "[Overview of Configuring Keystores for Message Protection](#)".
 - Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. Store the service's private key in the keystore for decrypting the message, and the CA root certificate.
 - Store the password for the decryption key in the credential store, as described in "[Adding Keys and User Credentials to Configure the Credential Store](#)". Use `keystore.enc.csf.key` as the key name.
 - Override the `keystore.enc.csf.key` server-side configuration property, as described in "[Overview of Policy Configuration Overrides](#)".

17.190 oracle/ wss_saml_bearer_or_username_token_sha256_service_policy

The `oracle/wss_saml_bearer_or_username_token_sha256_service_policy` enforces one authentication policy, based on whether the client uses a SAML bearer or username token.

Display Name: WSSecurity SAML Token Bearer or WSSecurity UserName Token Sha256 Service Policy

Category: Security

Description

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.

By default, SAML Bearer token is expected to be signed with an enveloped signature using RSA with SHA256 signature method. This policy can be applied to any SOAP-based endpoint.

To protect against replay attacks, the assertion provides the option to require nonce and 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.

Assertions (OR Group)

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_service_template](#)
- [oracle/wss_username_token_service_template](#)

The assertions are advertised in the WSDL.

17.191 oracle/ wss_saml_token_bearer_identity_switch_client_policy

The 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.

Display Name: Wss SAML Token Bearer Identity Switch Client Policy

Category: Security

Description

The SAML token with confirmation method Bearer is created automatically. This policy can be attached to any SOAP-based client.

Assertion

This policy contains the following assertion:

oracle/wss_saml_token_bearer_client_template

See "[oracle/wss_saml_token_bearer_client_template](#)" for more information about the assertion.

Configuration

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](#)" for more information about the assertion.

Settings

See [Table 18-54](#).

Configuration Properties

See [Table 18-55](#).

17.192 oracle/ wss_saml_token_bearer_identity_switch_sha256_client_policy

The oracle/wss_saml_token_bearer_identity_switch_sha256_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.

Display Name: Wss SAML Token Bearer Identity Switch Sha256 Client Policy

Category: Security

Description

The SAML token with confirmation method Bearer is created automatically and is by default signed with an enveloped signature using RSA with SHA256 signature method. The policy also verifies that the transport protocol provides SSL message protection. This policy can be attached to any SOAP-based client.

Assertion

This policy contains the following assertion:

oracle/wss_saml_token_bearer_client_template

See "[oracle/wss_saml_token_bearer_client_template](#)" for more information about the assertion.

Configuration

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](#)" for more information about the assertion.

Settings

See [Table 18-54](#).

Configuration Properties

See [Table 18-55](#).

17.193 oracle/ wss_saml_token_bearer_over_ssl_sha256_client_policy

The oracle/wss_saml_token_bearer_over_ssl_sha256_client_policy includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method Bearer is created automatically.

Display Name: Wss SAML Token (confirmation method as bearer) Over SSL Sha256 Client Policy

Category: Security

Description

The SAML token is signed using RSA with SHA256 signature method. The issuer name and subject name are provided either programmatically or through the current Java Authentication and Authorization Service (JAAS) subject. The policy also verifies that the transport protocol provides SSL message protection. This policy can be attached to any SOAP-based client.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml_token_bearer_over_ssl_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-59](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure SAML on the client side, as described in "[Configuring SAML Web Service Client at Design Time](#)".

17.194 oracle/ wss_saml_token_bearer_over_ssl_sha256_service_policy

The oracle/wss_saml_token_bearer_over_ssl_sha256_service_policy authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header. It accepts SAML tokens signed using RSA with SHA256 signature method.

Display Name: Wss SAML Token (confirmation method as bearer) Over SSL Sha256 Service Policy

Category: Security

Description

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.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml_token_bearer_over_ssl_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-60](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml.loginmodule` login module. See "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)" for more information. The SAML login module extracts the username from the verified token and passes it to the Authentication provider.
- Configure SAML and set up OPSS, as described in "[About SAML Configuration](#)".

17.195 oracle/wss_saml_token_bearer_service_policy

The oracle/wss_saml_token_bearer_service_policy authenticates users using credentials provided in SAML Bearer token in the WS-Security SOAP header. By default, SAML Bearer token is expected to be signed with an enveloped signature.

Display Name: WSSecurity SAML Token Bearer Service Policy

Category: Security

Description

This policy can be applied to any SOAP-based endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml_token_bearer_service_template](#)

This assertion is advertised in the WSDL.

17.196 oracle/wss_saml_token_bearer_sha256_client_policy

The oracle/wss_saml_token_bearer_sha256_client_policy includes SAML Bearer tokens in outbound SOAP request messages.

Display Name: Wss SAML Token (confirmation method as bearer) Sha256 Client Policy

Category: Security

Description

The SAML token with confirmation method Bearer is created automatically and is by default signed with an enveloped signature using RSA with SHA256 signature method.

The issuer name and subject name are provided either programmatically or through the current Java Authentication and Authorization Service (JAAS) subject.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml_token_bearer_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-59](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure SAML on the client side, as described in "[Configuring SAML Web Service Client at Design Time](#)".

17.197 oracle/wss_saml_token_bearer_sha256_service_policy

The oracle/wss_saml_token_bearer_sha256_service_policy authenticates users using credentials provided in SAML Bearer token in the WS-Security SOAP header. SAML Bearer token is expected to be signed with an enveloped signature using RSA with SHA256 signature method.

Display Name: WSSecurity SAML Token Bearer Sha256 Service Policy

Category: Security

Description

This policy can be applied to any SOAP-based endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml_token_bearer_service_template](#)

This assertion is advertised in the WSDL.

17.198 oracle/binding_oes_authorization_policy

The oracle/binding_oes_authorization_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.

Display Name: Fine-grained authorization using Oracle Entitlements Server

Category: Security

Description

This policy should follow an authentication policy where the subject is established. You can attach this policy to any SOAP endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/binding_oes_authorization_template](#)

This assertion is not advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-107](#). For more information, see "[Overriding Policy Configuration Properties](#)".

17.199 oracle/binding_oes_masking_policy

The oracle/binding_oes_masking_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.

Display Name: Response masking using Oracle Entitlements Server

Category: Security

Description

This policy should follow an authentication policy where the subject is established. You can attach this policy to any SOAP endpoint.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/binding_oes_masking_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-107](#). For more information, see "[Overriding Policy Configuration Properties](#)".

17.200 oracle/component_oes_authorization_policy

The oracle/component_oes_authorization_policy does user authorization based on the policy defined in Oracle Entitlements Server (OES).

Display Name: SCA Component fine-grained authorization using Oracle Entitlements Server

Category: Security

Description

This policy does user authorization based on the policy defined in Oracle Entitlements Server (OES)

17.201 oracle/jms_transport_client_policy

The JMS Transport Client Policy enables and configures support for SOAP over JMS transport for web service clients.

Display Name: JMS Transport Client Policy

Category: SOAP Over JMS Transport

Description

Enables and configures support for SOAP over JMS transport for web service clients.



Note:

This policy cannot be duplicated, and the assertion template associated with this template is not available for generating new policies.

This policy is not supported for Java EE (WebLogic) web services.

Configuration

Table 17-48 lists the configuration properties that you can override for SOAP over JMS transport clients.

Table 17-48 Configuration Properties for oracle/jms_transport_client_policy

Name	Description	Default	Required?
destination.name	JNDI name of the destination queue or topic.	com.oracle.webservices.api.jms.RequestQueue	Required
destination.type	Destination type. Valid values include: com.oracle.webservices.api.jms.JMSDestinationType.QUEUE or com.oracle.webservices.api.jms.JMSDestinationType.TOPIC. This value defaults to QUEUE.	QUEUE	Required
jms.header.property	JMS header properties. Each property is specified using name-value pairs, separated by semicolons (;). For example: name1=value1&...&nameN=valueN.	None	Optional
jms.message.property	JMS message properties. Each property is specified using name-value pairs, separated by semicolons (;). For example: name1=value1&...&nameN=valueN.	None	Optional
jndi.connection.factory.name	JNDI name of the connection factory that is used to establish a JMS connection.	com.oracle.webservices.jms.ConnectionFactory	Required

Table 17-48 (Cont.) Configuration Properties for oracle/jms_transport_client_policy

Name	Description	Default	Required?
jndi.context.parameters	JNDI properties. Each property is specified using name-value pairs, separated by semicolons (;). For example: name1=value1&...&nameN=valueN. The properties are added to the <code>java.util.Hashtable</code> sent to the <code>InitialContext</code> constructor for the JNDI provider.	None	Optional
jndi.initial.context.factory	Name of the initial context factory class used for JNDI lookup. This value maps to the <code>java.naming.factory.initial</code> property.	<code>weblogic.jndi.WLInitialContextFactory</code>	Required
jndiurl	JNDI provider URL. This value maps to the <code>java.naming.provider.url</code> property.	<code>t3://localhost:7001</code>	Required
message.type	Message type to use with the request message. Valid values are <code>com.oracle.webservices.api.jms.JMSMessageType.BYTES</code> and <code>com.oracle.webservices.api.jms.JMSMessageType.TEXT</code> . This value defaults to <code>BYTES</code> . For more information, see "Configuring the JMS Message Type" in <i>Developing JAX-WS Web Services for Oracle WebLogic Server</i> .	<code>BYTES</code>	Required
priority	JMS priority associated with the request and response message. Specify this value as a positive Integer from 0, the lowest priority, to 9, the highest priority. The default value is 0.	0	Required

Table 17-48 (Cont.) Configuration Properties for oracle/jms_transport_client_policy

Name	Description	Default	Required?
reply.to.name	<p>JNDI name of the JMS destination to which the response message is sent.</p> <p>For a two-way operation, a temporary response queue is generated by default. Using the default temporary response queue minimizes the configuration that is required. However, in the event of a server failure, the response message may be lost.</p> <p>This property enables the client to use a previously defined, "permanent" queue or topic rather than use the default temporary queue or topic, for receiving replies. For more information about configuring the JMS response queue, see "Configuring the Response Queue" in <i>Developing JAX-WS Web Services for Oracle WebLogic Server</i>.</p> <p>The value maps to the <code>JMSReplyTo</code> JMS header in the request message.</p>	None	Optional
target.service	<p>Port component name of the web service. This value is used by the service implementation to dispatch the service request. If not specified, the service name from the WSDL or <code>@javax.jws.WebService</code> annotation is used.</p> <p>This value maps to the <code>SOAPJMS_targetService</code> JMS message property.</p>	None	Optional
time.to.live	<p>Lifetime, in milliseconds, of the request message. A value of 0 indicates an infinite lifetime.</p> <p>On the service side, <code>timeToLive</code> also specifies the expiration time for each MDB transaction.</p>	180000	Required
reference.priority	See " reference.priority ".	None	Optional

17.202 oracle/jms_transport_service_policy

The JMS Transport Service Policy overrides configuration properties for SOAP over JMS transport for web services.

Display Name: JMS Transport Service Policy

Category: SOAP Over JMS Transport

Description



Note:

This policy cannot be duplicated, and the assertion template associated with this template is not available for generating new policies.

This policy is not supported for Java EE (WebLogic) web services.

Configuration

[Table 17-49](#) lists the configuration properties that you can override for SOAP over JMS transport for web services.

Table 17-49 Configuration Properties for oracle/jms_transport_service_policy

Name	Description	Default	Required?
binding.version	Version of the SOAP JMS binding. This value must be set to SOAP_JMS_1.0 for this release, which equates to com.oracle.webservices.api.jms.JMSBindingVersion.SOAP_JMS_1_0. This value maps to the SOAPJMS_bindingVersion JMS message property	SOAP_JMS_1.0	Required
delivery.mode	Delivery mode indicating whether the request message is persistent. Valid values are com.oracle.webservices.api.jms.DeliveryMode.PERSISTENT and com.oracle.webservices.api.jms.DeliveryMode.NON_PERSISTENT.	PERSISTENT	Required
enable.http.wSDL.access	Boolean flag that specifies whether to publish the WSDL through HTTP.	true	Optional
run.as.principal	Principal used to run the listening MDB.	None	Optional
run.as.role	Role used to run the listening MDB.	None	Optional
mdb.per.destination	Boolean flag that specifies whether to create one listening message-driven bean (MDB) for each requested destination. If set to false, one listening MDB is created for each web service port, and that MDB cannot be shared by other ports.	true	Optional

Table 17-49 (Cont.) Configuration Properties for oracle/jms_transport_service_policy

Name	Description	Default	Required?
activation.config	Activation configuration properties passed to the JMS provider. Each property is specified using name-value pairs, separated by semicolons (;). For example: name1=value1&...&nameN=valueN. For a list of activation configuration properties that are supported by this property, see "Summary of JMS Transport Configuration Properties" in <i>Developing JAX-WS Web Services for Oracle WebLogic Server</i> .	None	Optional
destination.name	JNDI name of the destination queue or topic.	com.oracle.webservice.s.api.jms.RequestQueue	Required
destination.type	Destination type. Valid values include: com.oracle.webservices.api.jms.JMSDestinationType.QUEUE or com.oracle.webservices.api.jms.JMSDestinationType.TOPIC. This value defaults to QUEUE.	QUEUE	Required
jms.header.property	JMS header properties. Each property is specified using name-value pairs, separated by semicolons (;). For example: name1=value1&...&nameN=valueN.	None	Optional
jms.message.property	JMS message properties. Each property is specified using name-value pairs, separated by semicolons (;). For example: name1=value1&...&nameN=valueN.	None	Optional
jndi.connection.factory.name	JNDI name of the connection factory that is used to establish a JMS connection.	com.oracle.webservice.s.jms.ConnectionFactory	Required
jndi.context.parameters	JNDI properties. Each property is specified using name-value pairs, separated by semicolons (;). For example: name1=value1&...&nameN=valueN. The properties are added to the java.util.Hashtable sent to the InitialContext constructor for the JNDI provider.	None	Optional
jndi.initial.context.factory	Name of the initial context factory class used for JNDI lookup. This value maps to the java.naming.factory.initial property.	weblogic.jndi.WLInitialContextFactory	Required

Table 17-49 (Cont.) Configuration Properties for oracle/jms_transport_service_policy

Name	Description	Default	Required?
jndiurl	JNDI provider URL. This value maps to the <code>java.naming.provider.url</code> property.	t3://localhost:7001	Required

17.203 oracle/no_jms_transport_client_policy

The `oracle/no_jms_transport_client_policy` is a no behavior policy, when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached SOAP over JMS transport client policy at a higher scope.

Display Name: No Jms Transport Client Policy

Category: SOAP Over JMS Transport

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no_behavior policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-50](#) lists the configuration property that you can override for the no behavior policy.

Table 17-50 Configuration Property for oracle/no_jms_transport_client_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.204 oracle/no_jms_transport_service_policy

The oracle/no_jms_transport_service_policy is a no behavior policy, when directly attached to an endpoint or globally attached at a lower scope, effectively disables a globally attached SOAP over JMS transport service policy at a higher scope.

Display Name: No Jms Transport Client Policy

Category: SOAP Over JMS Transport

Description

For details about using this no behavior policy, see "[Disabling a Globally Attached Policy](#)".

Note:

Please note the following:

- This no behavior policy cannot be duplicated.
- The assertion template associated with this no behavior policy is not available for generating new policies.
- This no_behavior policy is not supported for Java EE (WebLogic) web services.

Assertion

All no behavior policies use the same no behavior assertion. An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. For information about restoring the repository, see "[Rebuilding the OWSM Repository](#)".

Configuration

[Table 17-51](#) lists the configuration property that you can override for the no behavior policy.

Table 17-51 Configuration Property for oracle/no_jms_transport_service_policy

Name	Description	Default	Required?
reference.priority	See " reference.priority ".	None	Optional

17.205 oracle/ http_oauth2_token_over_ssl_salesforce_jwt_client_policy

You can attach the oracle/http_oauth2_token_over_ssl_salesforce_jwt_client_policy to client applications that need to obtain an Access Token from the Salesforce OAuth2 server in order to access certain resources.

Display Name: HTTP OAuth2 Token Over SSL Salesforce JWT Client Policy

Category: Security

Description

The `oracle/http_oauth2_token_over_ssl_salesforce_jwt_client_policy` can be attached to client applications that need to obtain an Access Token from the Salesforce OAuth2 server in order to access certain resources. It has been customized with certain properties that are required by OWSM to generate a JWT token that would be acceptable by the Salesforce OAuth2 server in order to issue an Access Token in its return.

The policy 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 client.

Assertion

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](#)" for more information about the assertion.

Configuration

The `oracle/http_oauth2_token_over_ssl_salesforce_jwt_client_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 OAuth2 access token in the HTTP header. The AT is obtained from the Salesforce OAuth2 server.

You can override the following properties when you attach the policy:

- `csf-key`
- `oauth2.client.csf.key`
- `audience.uri`

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](#)" for information about the assertion attributes that you can configure.

See "[Overriding Policy Configuration Properties](#)" 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.

Settings

See [Table 18-30](#).

Configuration Properties

See [Table 18-27](#).

17.206 oracle/multi_token_rest_access_service_policy

The oracle/multi_token_rest_access_service_policy allows access to endpoint with anonymous subject when there is no security token in the request. It also masks 403 response from service if security token is not present in the request.

Display Name: Multi Token RESTful Access Service Policy

Category: Security

Description

This policy enforces exactly 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 v2.0 *Bearer* token in the HTTP header—Extracts SAML 2.0 *Bearer* assertion in the HTTP header.
- JWT token security —Extracts username from the JWT token in the HTTP header.
- HTTP OAM security (disabled by default) —Verifies that the OAM agent has authenticated user and establishes identity. (Provides non-SSL OAM protection on the server-side only.)

If there is no security token in the request, this policy bypasses the request to the service and establishes an anonymous subject in the context, which can be later verified by the service itself.

If there is a security token associated with the request, then the policy establishes a valid non-anonymous subject in the context.

If service sends 403 response and anonymous subject was established, OWSM masks response code to 401 Unauthorized and adds a WWW-Authenticate challenge header in the response.

If there is a non-anonymous subject established in the context and the service still returns a 403 Forbidden response, OWSM passes on the 403 Forbidden response.

Assertions (OR Group)

This policy contains assertions that are based on 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](#)
- [oracle/http_saml20_token_bearer_service_template](#)
- [oracle/http_oam_token_service_template](#)
- No Behavior Assertion: An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. See [Rebuilding the OWSM Repository](#).

The oracle/http_saml20_token_bearer_service_template policy assertions are advertised.

The wss_http_token_service_template assertions are not advertised in the WSDL.

**Note:**

Advertisement of policy assertions in a WADL file is not supported. The Advertised option has no effect when this policy is attached to a RESTful web service.

Configuration

Table 17-52 lists the configuration property that you can use to configure this policy.

Table 17-52 Configuration Property for oracle/multi_token_rest_access_service_policy

Name	Description	Default	Required?
<code>anonymous.access</code>	See anonymous.access .	True	Optional

17.207 oracle/multi_token_rest_access_over_ssl_service_policy

The oracle/multi_token_rest_access_over_ssl_service_policy allows access to endpoint over SSL with anonymous subject when there is no security token in the request. Also, masks 403 response from service if security token is not present in request.

Display Name: Multi Token RESTful Access Over SSL Service Policy

Category: Security

Description

This policy enforces exactly 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 v2.0 *Bearer* token in the HTTP header—Extracts SAML 2.0 *Bearer* assertion in the HTTP header.
- JWT token security —Extracts username from the JWT token in the HTTP header.
- HTTP OAM security (disabled by default) —Verifies that the OAM agent has authenticated user and establishes identity. (Provides non-SSL OAM protection on the server-side only.)

If there is no security token in the request, this policy bypasses the request to the service and establishes an anonymous subject in the context, which can be later verified by the service itself.

If there is a security token associated with the request, then the policy establishes a valid non-anonymous subject in the context.

If service sends 403 response and anonymous subject was established, OWSM masks response code to 401 Unauthorized and adds a WWW-Authenticate challenge header in the response.

If there is a non-anonymous subject established in the context and the service still returns a 403 Forbidden response, OWSM passes on the 403 Forbidden response.

This policy expects that the service invocation to be done over SSL.

Assertions (OR Group)

This policy contains assertions that are based on 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](#)
- [oracle/http_saml20_token_bearer_service_template](#)
- [oracle/http_oam_token_service_template](#)
- No Behavior Assertion: An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. See [Rebuilding the OWSM Repository](#).

The `oracle/http_saml20_token_bearer_service_template` policy assertions are advertised.

The `wss_http_token_service_template` assertions are not advertised in the WSDL.

 **Note:**

Advertisement of policy assertions in a WADL file is not supported. The Advertised option has no effect when this policy is attached to a RESTful web service.

Configuration

[Table 17-53](#) lists the configuration properties that you can use to configure this policy.

Table 17-53 Configuration Property for `oracle/multi_token_rest_access_over_ssl_service_policy`

Name	Description	Default	Required?
<code>anonymous.access</code>	See anonymous.access .	True	Optional

17.208 [oracle/http_anonymous_rest_service_policy](#)

The `oracle/http_anonymous_rest_service_policy` allows access to endpoint with anonymous subject in context.

Display Name: Http Anonymous RESTful Service Policy

Category: Security

Description

This policy is an extension of functionality provided by `no_authentication_service_policy`.

If there is a security token or there is no security token in the request, then in both the cases this policy bypasses the request to the service and establishes an anonymous subject in the context.

Assertions (OR Group)

This policy contains assertions that are based on the following assertion templates as an OR group—meaning any one of the tokens can be sent by the client:

- **No Behavior Assertion:** An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. See [Rebuilding the OWSM Repository](#).

 **Note:**

Advertisement of policy assertions in a WADL file is not supported. The Advertised option has no effect when this policy is attached to a RESTful web service.

Configuration

[Table 17-54](#) lists the configuration property that you can use to configure this policy.

Table 17-54 Configuration Property for `oracle/http_anonymous_rest_service_policy`

Name	Description	Default	Required?
<code>anonymous.access</code>	See anonymous.access .	True	Optional

17.209 `oracle/http_anonymous_rest_over_ssl_service_policy`

The `oracle/http_anonymous_rest_over_ssl_service_policy` allows access to endpoint over SSL with anonymous subject in context.

Display Name: Http Anonymous RESTful Over SSL Service Policy

Category: Security

Description

This policy is an extension of functionality provided by `no_authentication_service_policy`.

If there is a security token or there is no security token in the request, then in both the cases this policy bypasses the request to the service and establishes an anonymous subject in the context.

This policy expects that the service invocation to be done over SSL.

Assertions (OR Group)

This policy contains assertions that are based on the following assertion templates as an OR group—meaning any one of the tokens can be sent by the client:

- **No Behavior Assertion:** An assertion template is not provided for the no behavior assertion. For that reason, it is important that you do not delete the no behavior policies. To recreate them you will need to restore the OWSM repository with the original policies. See [Rebuilding the OWSM Repository](#).

 **Note:**

Advertisement of policy assertions in a WADL file is not supported. The Advertised option has no effect when this policy is attached to a RESTful web service.

Configuration

Table 17-55 lists the configuration property that you can use to configure this policy.

Table 17-55 Configuration Property for oracle/http_anonymous_rest_over_ssl_service_policy

Name	Description	Default	Required?
anonymous.access	See anonymous.access .	True	Optional

17.210 oracle/ http_oauth2_token_over_ssl_google_jwt_client_policy

The oracle/http_oauth2_token_over_ssl_google_jwt_client_policy can be used by clients that need to access Google APIs.

Display Name: HTTP oauth2 token over ssl google jwt client policy

Category: Security

Description

The oracle/http_oauth2_token_over_ssl_google_jwt_client_policy can be attached to client applications that need to access Google APIs. This policy enables a client to obtain the OAuth2 access token from the Google OAUTH2 server. This token is set in the HTTP Header of the client request issued to access the corresponding Google API. It has been customized with certain properties that are required by OWSM to generate a JWT token that would be acceptable by the Google OAUTH2 server.

The policy 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 client.

Assertion

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](#)".

Configuration

You can attach oracle/http_oauth2_token_over_ssl_google_jwt_client_policy to client applications that need to access Google APIs.

Here is a list of the respective properties to be used:

- oracle/oauth2_config_client_policy
 - token.uri = https://accounts.google.com/o/oauth2/token
- oracle/http_oauth2_token_over_ssl_google_jwt_client_policy.
 - subject.precedence = false (default)
 - oauth2.client.csf.key = google-service-credential (See [Adding Keys and User Credentials to Configure the Credential Store](#))

- include.client.credentials = false (default)
- issuer.name = \${client.id} (determined at runtime)
- audience.uri = https://accounts.google.com/o/oauth2/token
- keystore.sig.csf.key = privatekey (See [Configuring the OWSM Keystore](#))
- custom.jwt.claims = scope=api1 api2 apiN (See

OWSM sends request to the Google OAuth2 server with the JWT token. The Request must look like this -

Post /o/oauth2/token

host: https://accounts.google.com

Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer&assertion=<JWT Token>

Note:

The "audience.uri" and "issuer.name" have default values provided in the policy.

Note:

OWSM generates a new JWT token for the outgoing request, requesting a new Access Token, if there is a change in the value of the custom.jwt.claims. In other words the cached JWT token cannot be used if there is a change in this value.

OWSM sends a request to the Google OAuth2 server with the JWT which is generated by the WSM agent for authentication. If the JWT token is authenticated successfully by the Google server, an Access Token is sent in the response. The default validity of the token is 1 hour.

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](#)" for information about the assertion attributes that you can configure.

See "[Overriding Policy Configuration Properties](#)" 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.

Settings

See [Table 18-30](#).

Configuration Properties

See [Table 18-27](#).

17.211 oracle/ wss_saml20_token_bearer_over_ssl_notimestamp_client_policy

The oracle/wss_saml20_token_bearer_over_ssl_notimestamp_client_policy includes SAML tokens in outbound SOAP request messages, and verifies that the transport protocol provides SSL message protection.

Display Name: Wss SAML V2.0 Token (confirmation method as bearer) Over SSL With No Timestamp Client Policy

Category: Security

Description

The SAML token with confirmation method *Bearer* is created automatically. This policy can be attached to any SOAP-based client. Timestamp is not added to the message.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml20_token_bearer_over_ssl_client_template](#)

This assertion is advertised.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-62](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- 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 Using SAML Policies](#)" for additional considerations.

Design Time Considerations

At design time:

- Override configuration settings, as described in "[About Overriding Client Policy Configuration Properties at Design Time](#)".
- Configure SAML on the client side, as described in "[Configuring SAML Web Service Client at Design Time](#)".

17.212 oracle/ wss_saml20_token_bearer_over_ssl_notimestamp_service_policy

The oracle/wss_saml20_token_bearer_over_ssl_notimestamp_service_policy authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header, and verifies that the transport protocol provides SSL message protection.

Display Name: Wss SAML V2.0 Token (confirmation method as bearer) Over SSL With No Timestamp Service Policy

Category: Security

Description

The credentials in the SAML token are authenticated against a SAML login module. This policy can be enforced on any SOAP-based endpoint.

The SAML login module extracts the username from the verified token and passes it to the Authentication provider. Timestamp should not be present in the incoming message.

Assertion

This policy contains an assertion that is based on the following assertion template, which defines the settings and configuration properties for the policy:

- [oracle/wss_saml20_token_bearer_over_ssl_service_template](#)

This assertion is advertised in the WSDL.

Configuration

To configure the policy:

- Override the configuration properties defined in [Table 18-63](#). For more information, see "[Overriding Policy Configuration Properties](#)".
- Configure one-way or two-way SSL, as described in "[Configuring One-Way SSL on WebLogic Server](#)" or "[Configuring Two-Way SSL for a Web Service Client](#)", respectively.
- Specify a value for `propagate.identity.context`, as described in "[Overriding Policy Configuration Properties](#)". The `propagate.identity.context` property defaults to a value of blank. For additional considerations, see "[Propagating Identity Context Using SAML Policies](#)".
- Add an Authentication provider to the active security realm for the WebLogic domain in which the web service is deployed, as described in "[Supported Authentication Providers in WebLogic Server](#)".
- Configure the `saml2.loginmodule` login module, as described in "[Configuring the SAML and SAML2 Login Modules Using Fusion Middleware Control](#)". The SAML login module extracts the username from the verified token and passes it to the provider.
- Configure SAML and set up OPSS, as described in "[About SAML Configuration](#)".

17.213 oracle/http_oauth2_token_idcs_client_policy

The oracle/http_oauth2_token_idcs_client_policy includes the OAuth2 access token in the HTTP header. The access token is obtained from the IDCS 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.

Display Name: HTTP OAuth2 Token IDCS Client Policy

Category: Security

Description

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 true 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.

The set.client.ID is set to false by default. If it is set to true, OWSM sends client ID to OAuth2 provider in access token request as query param.

Assertion

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](#)" for more information about the assertion.

Configuration

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.

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
 - set.client.id

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](#)" for information about the assertion attributes that you can configure.

You attach this policy and the `oracle/oauth2_config_client_policy` to the client application. The `token.uri` property of the required `oracle/oauth2_config_client_policy` policy specifies the OAuth2 server. It also has the `oauth2.client.csf.key` property.

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 `true` 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 "[Setting the Permission Using WSIdentityPermission](#)".

By default, the `oracle/http_oauth2_token_idcs_client_policy` assertion content is defined as follows:

```

<orasp:http-oauth2-security xmlns:ns0="http://schemas.oracle.com/ws/2006/01/policy"
ns0:Silent="true" ns0:name="Http OAuth2 Security" ns0:Enforced="true"
ns0:category="security/authentication">
<orasp:auth-header orasp:mechanism="oauth2"/>
<orasp:bindings>
<orasp:Config orasp:name="HttpOAuth2Config" orasp:configType="declarative">
<orasp:PropertySet orasp:name="standard-security-properties">
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="subject.precedence">
<orasp:Value/>
<orasp:DefaultValue>true</orasp:DefaultValue>

    </orasp:Property>
    <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="csf.map"/>
    <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="csf-key">
    <orasp:Value/>
    </orasp:Property>
    <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="oauth2.client.csf.key">
    <orasp:Value/>
    <orasp:DefaultValue>NONE</orasp:DefaultValue>
    </orasp:Property>
    <orasp:Property orasp:type="boolean" orasp:contentType="optional"
orasp:name="federated.client.token">
    <orasp:Value/>
<orasp:DefaultValue>true</orasp:DefaultValue>
</orasp:Property>
<orasp:Property orasp:type="boolean" orasp:contentType="optional"
orasp:name="set.client.id">
<orasp:Value/>
<orasp:DefaultValue>>false</orasp:DefaultValue>
</orasp:Property>
<orasp:Property orasp:type="string" orasp:contentType="optional" orasp:name="scope">
<orasp:Value/>
</orasp:Property>
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="authz.code">
    <orasp:Value/>
  </orasp:Property>
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="redirect.uri">
    <orasp:Value/>
  </orasp:Property>
  <!-- Begin : properties needed for local token creation for end user-->
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="user.attributes">
    <orasp:Value/>
  </orasp:Property>
  <orasp:Property orasp:type="string" orasp:contentType="optional"
orasp:name="issuer.name">
    <orasp:Value/>
    <orasp:DefaultValue>www.oracle.com</orasp:DefaultValue>
  </orasp:Property>
  <orasp:Property orasp:type="boolean" orasp:contentType="optional"
orasp:name="oracle.oauth2.service">
    <orasp:Value/>
    <orasp:DefaultValue>>false</orasp:DefaultValue>
  </orasp:Property>
  <orasp:Property orasp:type="boolean" orasp:contentType="optional"
orasp:name="user.roles.include">

```

```

        <orawsp:Value/>
        <orawsp:DefaultValue>>false</orawsp:DefaultValue>false>
    </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>NONE>
    </orawsp:Property>
    <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="include.certificate">
        <orawsp:Value/>
        <orawsp:DefaultValue>>false</orawsp:DefaultValue>false>
    </orawsp:Property>
    <orawsp:Property orawsp:type="boolean" orawsp:contentType="optional"
orawsp:name="time.in.millis">
        <orawsp:Value/>
        <orawsp:DefaultValue>>true</orawsp:DefaultValue>true>
    </orawsp:Property>
    <!--End properties for local token creation for end user -->
</orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orasp:http-oauth2-security>

```

Settings

See [Table 18-30](#).

Configuration Properties

See [Table 18-27](#).

17.214 oracle/http_oauth2_token_over_ssl_idcs_client_policy

The `oracle/http_oauth2_token_over_ssl_idcs_client_policy` includes the OAuth2 access token in the HTTP header. The access token is obtained from the IDCS 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.

Display Name: HTTP OAuth2 Token Over SSL IDCS Client Policy

Category: Security

Description

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 true 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.

The `set.client.ID` is set to false by default. If it is set to true, OWSM sends client ID to OAuth2 provider in access token request as query param.

Assertion

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](#)" for more information about the assertion.

Configuration

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.

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
 - set.client.id

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](#)" for information about the assertion attributes that you can configure.

You attach this policy and the `oracle/oauth2_config_client_policy` to the client application. The `token.uri` property of the required `oracle/oauth2_config_client_policy` specifies the OAuth2 server. It also has the `oauth2.client.csf.key` property.

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 `true` 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 "[Setting the Permission Using WSIdentityPermission](#)".

By default, the `oracle/http_oauth2_token_over_ssl_idcs_client_policy` assertion content is defined as follows:

```
<orasp:http-oauth2-security xmlns:ns0="http://schemas.oracle.com/ws/2006/01/policy"
ns0:Silent="true" ns0:name="Http OAuth2 Security" ns0:Enforced="true"
ns0:category="security/authentication,security/msg-protection">
<orasp:auth-header orasp:mechanism="oauth2"/>
<>
<orawsp:bindings>
<orawsp:Config orawsp:name="HttpOAuth2Config" orawsp:configType="declarative">
<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="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>NONEorawsp:DefaultValue>NONE</orawsp:Property>
        <orawsp:Property orawsp:type="boolean" orawsp:contentType="optional"
orawsp:name="federated.client.token">
            <orawsp:Value/>
<orawsp:DefaultValue>trueorawsp:DefaultValue>true</orawsp:Property>
<orawsp:Property orawsp:type="boolean" orawsp:contentType="optional"
orawsp:name="set.client.id">
            <orawsp:Value/>
<orawsp:DefaultValue>>falseorawsp:DefaultValue>false</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>
        <!-- Begin : properties needed for local token creation for end user-->
        <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.comorawsp:DefaultValue>www.oracle.com</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>>falseorawsp:DefaultValue>false</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>
    <!--End properties for local token creation for end user -->
  </orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orawsp:http-oauth2-security>

```

Settings

See [Table 18-30](#).

Configuration Properties

See [Table 18-27](#).

17.215 oracle/ http_oauth2_token_identity_switch_over_ssl_idcs_client_policy

The oracle/http_oauth2_token_identity_switch_over_ssl_idcs_client_policy includes the OAuth2 access token in the HTTP header. The access token is obtained from the IDCS 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.

Display Name: HTTP Oauth2 Token Identity Switch Over SSL IDCS Client Policy

Category: Security

Description

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.

The `set.client.ID` is set to `false` by default. If it is set to `true`, OWSM sends client ID to OAuth2 provider in access token request as query param.

Assertion

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](#)" for more information about the assertion.

Configuration

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.

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` (It has a constant value of `false`)
 - `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
 - set.client.id

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](#)" for information about the assertion attributes that you can configure.

You attach this policy and the `oracle/oauth2_config_client_policy` to the client application. The `token.uri` property of the required `oracle/oauth2_config_client_policy` specifies the OAuth2 server. It also has the `oauth2.client.csf.key` property.

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 "[Setting the Permission Using WSIdentityPermission](#)".

By default, the `oauth2_token_identity_switch_over_ssl_idcs_client` assertion content is defined as follows:

```
<orasp:http-oauth2-security xmlns:ns0="http://schemas.oracle.com/ws/2006/01/policy"
ns0:Silent="true" ns0:name="Http OAuth2 Security" ns0:Enforced="true"
ns0:category="security/authentication,security/msg-protection">
<orasp:auth-header orasp:mechanism="oauth2"/>
<>
<orawsp:bindings>
<orawsp:Config orawsp:name="HttpOAuth2Config" orawsp:configType="declarative">
<orawsp:PropertySet orawsp:name="standard-security-properties">
  <orawsp:Property orawsp:type="string" orawsp:contentType="optional"
orawsp:name="subject.precedence">
<orawsp:Value/>
<orawsp:DefaultValue>false</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="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>NONEorawsp:DefaultValue>NONE</orawsp:Property>
        <orawsp:Property orawsp:type="boolean" orawsp:contentType="optional"
orawsp:name="federated.client.token">
        <orawsp:Value/>
<orawsp:DefaultValue>trueorawsp:DefaultValue>true</orawsp:Property>
<orawsp:Property orawsp:type="boolean" orawsp:contentType="optional"
orawsp:name="set.client.id">
<orawsp:Value/>
<orawsp:DefaultValue>>falseorawsp:DefaultValue>false</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>
        <!-- Begin : properties needed for local token creation for end user-->
        <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.comorawsp:DefaultValue>www.oracle.com</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>>falseorawsp:DefaultValue>false</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: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>
<!--End properties for local token creation for end user -->
</orawsp:PropertySet>
</orawsp:Config>
</orawsp:bindings>
</orawsp:http-oauth2-security>

```

Settings

See [Table 18-30](#).

Configuration Properties

See [Table 18-27](#).

Oracle Web Services Manager Predefined Assertion Templates

This topic describes the predefined assertion templates defined for the current release. Use the predefined assertion templates to construct your own policies or clone to create new policies.

 **Note:**

The predefined policies and assertion templates distributed with the current release are read only. You must copy the policy or assertion template before modifying it. You also have the option of configuring 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](#)".

For a detailed description of the configuration settings in the tables, see "[Assertion Template Settings for Oracle Web Services](#)".

For a detailed description of the configuration properties listed in the tables, see [Assertion Template Configuration Properties for Oracle Web Services](#). For details on how to edit the configuration properties, see "[Editing the Configuration Properties in an Assertion Template](#)". For information about overriding policies, see "[Overview of Policy Configuration Overrides](#)".

- [Authentication Only Assertion Templates](#)
- [Message-Protection Only Assertion Templates](#)
- [Message Protection and Authentication Assertion Templates](#)
- [Oracle Entitlements Server \(OES\) Integration Templates](#)
- [PII Assertion Templates](#)
- [WS-Trust Assertion Templates](#)
- [Authorization Assertion Templates](#)
- [Management Assertion Templates](#)

18.1 Authentication Only Assertion Templates

This table summarizes the assertion templates that enforce authentication only, and indicates whether the token is inserted at the transport layer or SOAP header.

Table 18-1 Authentication Only Assertion Templates

Client Template	Service Template	Authenticatio n Transport	Authenticatio n SOAP	Authenticatio n REST	Message Protection Transport	Message Protection SOAP
N/A	oracle/ http_oam_token_ service_template	No	No	Yes	No	No
oracle/ http_saml20_toke n_bearer_client_t emplate	oracle/ http_saml20_toke n_bearer_service_ _template	No	No	Yes	Yes	No
oracle/ http_spnego_toke n_client_template	oracle/ http_spnego_toke n_service_templat e	No	No	Yes	Yes	No
oracle/ wss_http_token_c lient_template	oracle/ wss_http_token_s ervice_template	Yes	No	No	No	No
oracle/ wss_username_to ken_client_templa te	oracle/ wss_username_to ken_service_templ ate	No	Yes	No	No	No
oracle/ wss10_saml_toke n_client_template	oracle/ wss10_saml_toke n_service_templat e	No	Yes	No	No	No
oracle/ wss10_saml20_to ken_client_templa te	oracle/ wss10_saml20_to ken_service_templ ate	No	Yes	No	No	No
oracle/ wss11_kerberos_t oken_client_templ ate	oracle/ wss11_kerberos_t oken_service_templ ate	No	Yes	No	No	No
oracle/ http_oauth2_toke n_client_template		No	Yes	No	No	No
oracle/ http_oauth2_toke n_over_ssl_client_ _template	-	No	Yes	No	No	No
oracle/ oauth2_config_cli ent_template	-	No	Yes	No	No	No
oracle/ http_jwt_token_cli ent_template	oracle/ http_jwt_token_se rvice_template	No	Yes	No	No	No
oracle/ http_jwt_token_ov er_ssl_client_tem plate	oracle/ http_jwt_token_ov er_ssl_service_te mplate	No	Yes	No	No	No

18.2 Message-Protection Only Assertion Templates

Table 18-2 summarizes the assertion templates that enforce message protection only, and indicates whether the token is inserted at the transport layer or SOAP header.

Table 18-2 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

18.3 Message Protection and Authentication Assertion Templates

Table 18-3 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 18-3 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_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

Table 18-3 (Cont.) Message Protection and Authentication Assertion Templates

Client Template	Service Template	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss_username_token_over_ssl_client_template	oracle/wss_username_token_over_ssl_service_template	No	Yes	Yes	No
oracle/wss10_saml_hoken_with_message_protection_client_template	oracle/wss10_saml_hoken_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
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

Table 18-3 (Cont.) Message Protection and Authentication Assertion Templates

Client Template	Service Template	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss11_x509_token_with_message_protection_client_template	oracle/wss11_x509_token_with_message_protection_service_template	No	Yes	No	Yes
oracle/wss11_username_token_derivedkey_with_message_protection_signature_only_client_template	The service assertion includes both signature and encryption parts. The service assertion uses XOR method to process the request.	No	No	Yes	No
oracle/wss11_username_token_derivedkey_with_message_protection_encryption_only_client_template	The service assertion includes both signature and encryption parts. The service assertion uses XOR method to process the request.	No	No	Yes	No

18.4 Oracle Entitlements Server (OES) Integration Templates

This topic summarizes the assertion templates that are used for OES integration.

It includes the following topics:

- [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.

18.5 PII Assertion Templates

This section summarizes the assertion template that is used for PII security.

[oracle/pii_security_template](#) provides simple role-based authorization for the request based on the authenticated subject at the SOAP binding level.

18.6 WS-Trust Assertion Templates

This section summarizes the WS-Trust assertion templates.

 **Note:**

In this release, you can use Fusion Middleware Control to directly edit the assertion template text, but the Settings and Configuration pages are not available.

- [oracle/sts_trust_config_client_template](#)
- [oracle/sts_trust_config_service_template](#)
- [oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template](#)
- [oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template](#)
- [oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template](#)
- [oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template](#)
- [oracle/wss11_sts_issued_saml_with_message_protection_client_template](#)

18.7 Authorization Assertion Templates

This topic summarizes assertion templates that are used for authorization. Each authorization assertion template must follow an authentication assertion template.

- [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.

18.8 Management Assertion Templates

This topic summarizes the management assertion templates.

[oracle/security_log_template](#) provides a logging assertion template that can be attached to any binding or component.

18.9 oracle/http_oam_token_service_template

This topic describes the `http_oam_token_service_template` assertion template.

Display Name: Http OAM Service Assertion Template

Category: Security

Type: http-oam-security

Description

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 18-4 lists the settings for the `http_oam_token_service_template` assertion template.

Table 18-4 http_oam_token_service_template Settings

Name	Default Value
Authentication Header	
Authentication Header—Mechanism	oam
Authentication Header—Header Name	None

Configuration

Table 18-5 lists the default configuration properties and the default settings for the `http_oam_token_service_template` assertion template.

Table 18-5 http_oam_token_service_template Configuration Properties

Name	Default Value	Type
<code>reference.priority</code>	None	Optional
<code>remote-user</code>	OAM_REMOTE_USER	Optional

18.10 oracle/http_saml20_token_bearer_client_template

This topic describes the `http_saml20_token_bearer_client_template` assertion template

Display Name: Http Saml Bearer V2.0 Token Client Assertion Template

Category: Security

Type: http-saml20-bearer-security

Description

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 18-6 lists the settings for the `http_saml20_token_bearer_client_template` assertion template.

Table 18-6 http_saml20_token_bearer_client_template Settings

Name	Default Value
Authentication Header	
Authentication Header—Mechanism	saml20-bearer
Authentication Header—Header Name	None

Configuration

Table 18-7 lists the configuration properties and the default settings for the `http_saml20_token_bearer_client_template` assertion template.

Table 18-7 http_saml20_token_bearer_client_template Configuration Properties

Name	Default Value	Type
<code>user.attributes</code>	None	Optional
<code>saml.issuer.name</code>	<code>www.oracle.com</code>	Optional
<code>user.roles.include</code>	<code>false</code>	Optional
<code>csf-key</code>	<code>basic.credentials</code>	Optional
<code>subject.precedence</code>	<code>true</code>	Optional
<code>saml.audience.uri</code>	None	Optional
<code>keystore.sig.csf.key</code>	None	Optional
<code>saml.envelope.signature.required</code>	<code>true</code>	Optional
<code>reference.priority</code>	None	Optional
<code>propagate.identity.context</code>	None	Optional
<code>auth.header.token.type</code>	<code>oit</code>	Optional

18.11 oracle/http_saml20_token_bearer_service_template

This topic describes the `http_saml20_token_bearer_service_template` assertion template.

Display Name: Http Saml Bearer V2.0 Token Service Assertion Template

Category: Security

Type: http-saml20-bearer-security

Description

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 18-6 for information about the settings.

Configuration

Table 18-63 lists the configuration properties and the default settings for the `http_saml20_token_bearer_service_template` assertion template.

Table 18-8 `http_saml20_token_bearer_service_template` Configuration Properties

Name	Default Value	Type
saml.trusted.issuers	None	Optional
saml.envelope.signature.required	true	Optional
reference.priority	None	Optional
propagate.identity.context	None	Optional
auth.header.token.type	oit	Optional

18.12 oracle/http_spnego_token_client_template

This topic describes the `http_spnego_token_client_template` assertion template.

Display Name: SPNEGO Token Client Assertion Template

Category: Security

Type: http-spnego-security

Description

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 18-9 lists the settings for the `http_spnego_token_client_template` assertion template.

Table 18-9 `http_spnego_token_client_template` Settings

Name	Default Value
Authentication Header	
Authentication Header—Mechanism	spnego
Authentication Header—Header Name	None

Configuration

Table 18-10 lists the configuration properties and the default settings for the `http_spnego_token_client_template` assertion template.

Table 18-10 `http_spnego_token_client_template` Configuration Properties

Name	Default Value	Type
service.principal.name	HOST/localhost@EXAMPLE.COM	Required

Table 18-10 (Cont.) http_spnego_token_client_template Configuration Properties

Name	Default Value	Type
keytab.location	None	Optional
caller.principal.name	None	Optional
credential.delegation	false	Required
role	ultimateReceiver	Constant
reference.priority	None	Optional

18.13 oracle/http_spnego_token_service_template

This topic describes the `http_spnego_token_service_template` assertion template.

Display Name: SPNEGO Token Service Assertion Template

Category: Security

Type: http-spnego-security

Description

The `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 18-9](#) for information about the settings.

Configuration

[Table 18-11](#) lists the configuration properties and the default settings for the `http_spnego_token_service_template` assertion template.

Table 18-11 http_spnego_token_service_template Configuration Properties

Name	Default Value	Type
role	ultimateReceiver	Constant
credential.delegation	false	Required
reference.priority	None	Optional

18.14 oracle/wss_http_token_client_template

This topic describes the `wss_http_token_client_template` assertion template.

Display Name: Wss HTTP Token client Assertion Template

Category: Security

Type: http-security

Description

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 18-12 lists the settings for the `wss_http_token_client_template` assertion template.

Table 18-12 `wss_http_token_client_template` Settings

Name	Default Value
Authentication Header	
Authentication Header—Mechanism	basic
Authentication Header—Header Name	None
Transport Layer Security	
Transport Layer Security	Disabled
Transport Layer Security—Mutual Authentication Required	Disabled
Transport Layer Security—Include Timestamp	Disabled

Configuration

Table 18-13 lists the configuration properties and the default settings for the `wss_http_token_client_template` assertion template.

Table 18-13 `wss_http_token_client_template` Configuration Properties

Name	Default Value	Type
<code>csf-key</code>	<code>basic.credentials</code>	Required
<code>role</code>	<code>ultimateReceiver</code>	Constant
<code>reference.priority</code>	None	Optional
<code>include-timestamp</code>	<code>false</code>	Optional

18.15 oracle/wss_http_token_service_template

This topic describes the `wss_http_token_service_template` assertion template.

Display Name: Wss HTTP Token service Assertion Template

Category: Security

Type: http-security

Description

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 18-12](#) for information about the settings.

Configuration

[Table 18-14](#) lists the configuration properties and the default settings for the `wss_http_token_service_template` assertion template.

Table 18-14 `wss_http_token_service_template` Configuration Properties

Name	Default Value	Type
realm	owsm	Constant
role	ultimateReceiver	Constant
reference.priority	None	Optional

18.16 oracle/wss_username_token_client_template

This topic describes the `wss_username_token_client_template` assertion template.

Display Name: Wss Username Token client Assertion Template

Category: Security

Type: wss-username-token

Description

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:

If you do not use a digest password, policies created using this template are not secure. You should use this assertion with plain text or no password 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 18-15](#) lists the settings for the `wss_username_token_client_template` assertion template.

Table 18-15 wss_username_token_client_template Settings

Name	Default Value
Username Token	
Password Type	plaintext
Creation Time Required	Disabled
Nonce Required	Disabled

Configuration

[Table 18-16](#) lists the configuration properties and the default settings for the `wss_username_token_client_template` assertion template.

Table 18-16 wss_username_token_client_template Configuration Properties

Name	Default Value	Type
csf-key	<code>basic.credentials</code>	Required
role	<code>ultimateReceiver</code>	Constant
csf.map	None	Optional
user.tenant.name	None	Optional
reference.priority	None	Optional
include-timestamp	<code>false</code>	Optional

18.17 oracle/wss_username_token_service_template

This topic describes the `wss_username_token_service_template` assertion template.

Display Name: Wss Username Token service Assertion Template

Category: Security

Type: wss-username-token

Description

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:

If you do not use a digest password, policies created using this template are not secure. You should use this assertion with plain text or no password 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 18-15](#) for information about the settings.

Configuration

[Table 18-17](#) lists the configuration properties and the default settings for the `wss_username_token_service_template` assertion template.

Table 18-17 `wss_username_token_service_template` Configuration Properties

Name	Default Value	Type
role	ultimateReceiver	Constant
reference.priority	None	Optional

18.18 oracle/wss10_saml_token_client_template

This topic describes the `wss10_saml_token_client_template` assertion template.

Display Name: Wss10 SAML Token client Assertion Template

Category: Security

Type: wss10-saml-token

Description

The `wss10_saml_token_client_template` assertion template includes SAML tokens in outbound SOAP request messages. The SAML token is created automatically.

Settings

[Table 18-18](#) lists the settings for the `wss10_saml_token_client_template` assertion template.

Table 18-18 `wss10_saml_token_client_template` Settings

Name	Default Value
SAML Token Type	
Version	1.1
Confirmation Type	sender-vouches
Name Identifier Format	unspecified

Configuration

[Table 18-19](#) lists the configuration properties and the default settings for the `wss10_saml_token_client_template` assertion template.

Table 18-19 wss10_saml_token_client_template Configuration Properties

Name	Default Value	Type
user.attributes	None	Optional
user.roles.include	false	Optional
saml.issuer.name	www.oracle.com	Optional
csf-key	basic.credentials	Optional
subject.precedence	true	Optional
saml.audience.uri	None	Optional
propagate.identity.context	None	Optional
reference.priority	None	Optional
include-timestamp	false	Optional

18.19 oracle/wss10_saml_token_service_template

This topic describes the `wss10_saml_token_service_template` assertion template.

Display Name: Wss10 SAML Token service Assertion Template

Category: Security

Type: wss10-saml-token

Description

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. See [Table 18-18](#) for information about the settings.

Configuration

[Table 18-20](#) lists the configuration properties and the default settings for the `wss10_saml_token_service_template` assertion template.

Table 18-20 wss10_saml_token_service_template Configuration Properties

Name	Default Value	Type
role	ultimateReceiver	Constant
saml.trusted.issuers	None	Optional
propagate.identity.context	None	Optional
reference.priority	None	Optional

18.20 oracle/wss10_saml20_token_client_template

This topic describes the `wss10_saml20_token_client_template` assertion template.

Display Name: Wss10 SAML V2.0 Token client Assertion Template

Category: Security

Type: wss10-saml-token

Description

The `wss10_saml20_token_client_template` assertion template includes SAML tokens in outbound SOAP request messages. The SAML token is created automatically.

Settings

[Table 18-21](#) lists the settings for the `wss10_saml20_token_client_template` assertion template.

Table 18-21 `wss10_saml20_token_client_template` Settings

Name	Default Value
SAML Token Type	
Version	2.0
Confirmation Type	sender-vouches
Name Identifier Format	unspecified

Configuration

[Table 18-22](#) 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 in an Assertion Template](#)".

For information about overriding policies, see "[Overview of Policy Configuration Overrides](#)".

Table 18-22 `wss10_saml20_token_client_template` Configuration Properties

Name	Default Value	Type
<code>user.attributes</code>	None	Optional
<code>user.roles.include</code>	false	Optional
<code>saml.issuer.name</code>	www.oracle.com	Optional
<code>csf-key</code>	basic.credentials	Optional
<code>subject.precedence</code>	true	Optional
<code>saml.audience.uri</code>	None	Optional
<code>propagate.identity.context</code>	None	Optional
<code>reference.priority</code>	None	Optional
<code>include-timestamp</code>	false	Optional

18.21 oracle/wss10_saml20_token_service_template

This topic describes the `wss10_saml20_token_service_template` assertion template.

Display Name: Wss10 SAML V2.0 Token service Assertion Template

Category: Security

Type: wss10-saml-token

Description

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. See [Table 18-21](#) for information about the settings.

Configuration

[Table 18-23](#) 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 in an Assertion Template](#)".

For information about overriding policies, see "[Overview of Policy Configuration Overrides](#)".

Table 18-23 `wss10_saml20_token_service_template` Configuration Properties

Name	Default Value	Type
role	<code>ultimateReceiver</code>	Constant
saml.trusted.issuers	None	Optional
propagate.identity.context	None	Optional
reference.priority	None	Optional

18.22 oracle/wss11_kerberos_token_client_template

This topic describes the `wss11_kerberos_token_client_template` assertion template.

Display Name: Wss11 Kerberos Token client Assertion Template

Category: Security

Type: kerberos-security

Description

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 18-24](#) lists the settings for the `wss11_kerberos_token_client_template` assertion template.

Table 18-24 `wss11_kerberos_token_client_template` Settings

Name	Default Value
Kerberos Token Type	
Kerberos Token Type	<code>gss-apreq-v5</code>
Derived Keys	Disabled

Configuration

[Table 18-25](#) lists the configuration properties and the default settings for the `wss11_kerberos_token_client_template` assertion template.

Table 18-25 `wss11_kerberos_token_client_template` Configuration Properties

Name	Default Value	Type
service.principal.name	<code>HOST/ localhost@EXAMPLE.COM</code>	Required
keytab.location	None	Optional
caller.principal.name	None	Optional
credential.delegation	<code>false</code>	Required
reference.priority	None	Optional

18.23 oracle/wss11_kerberos_token_service_template

This topic describes the `wss11_kerberos_token_service_template` assertion template.

Display Name: Wss11 Kerberos Token service Assertion Template

Category: Security

Type: kerberos-security

Description

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_kerberos_token_service_template` are identical to the client version of the assertion template. See [Table 18-24](#) for information about the settings.

Configuration

[Table 18-26](#) lists the configuration properties and the default settings for the `wss11_kerberos_token_service_template` assertion template.

Table 18-26 `wss11_kerberos_token_service_template` Configuration Properties

Name	Default Value	Type
<code>credential.delegation</code>	false	Required
<code>role</code>	ultimateReceiver	Constant
<code>reference.priority</code>	None	Optional

18.24 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 18-27](#) lists the settings for the `http_oauth2_token_client_template` assertion template.

Table 18-27 http_oauth2_token_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	<p>Authentication mechanism.</p> <p>Valid values include:</p> <ul style="list-style-type: none"> • basic—Client authenticates itself by transmitting the username and password. <p>Note: It is recommended that you configure SSL when using basic authentication.</p> <ul style="list-style-type: none"> • 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. 	<pre><orasp:auth-header orasp:mechanism="oauth2"/></pre>
Authentication Header—Header Name	Name of the authentication header.	None
Authentication Header—is-signed	Flag that specifies whether the token is signed.	<pre><orasp:auth-header orasp:is-signed="false"/></pre>
Authentication Header—is encrypted	Flag that specifies whether the token is encrypted.	<pre><orasp:auth-header orasp:is-encrypted="false"/></pre>

Configurations

[Table 18-28](#) lists the default configuration properties for the http_oauth2_token_client_template assertion template.

Table 18-28 http_oauth2_token_client_template Configuration Properties

Name	Description
audience.uri	<p>Audience restriction. The following conditions are supported:</p> <ul style="list-style-type: none"> • If this property is not set, the service URL is used as the audience URI • If this property is set to <code>NONE</code> (not case sensitive), then the audience URI is set to null. • If this property is set to a value other than <code>NONE</code>, then the audience URI is set to this value. <p>Default setting:</p> <pre><orawsp:Property orawsp:contentType="optional" orawsp:name="audience.uri" orawsp:type="string"> <orawsp:Value/> <orawsp:DefaultValue>NONE</orawsp:DefaultValue></pre>
authz.code	<p>Optional property for passing the authorization code for the 3-legged OAuth2 use case. (Not supported in this release.)</p> <p>Default setting:</p> <pre><orawsp:Property orawsp:contentType="optional" orawsp:name="authz.code" orawsp:type="string"> <orawsp:Value/></pre>
csf-key	<p>Credential store key that maps to a user name and password in the Oracle Platform Security Services (OPSS) identity store.</p> <p>Default setting:</p> <pre><orawsp:Property orawsp:type="string" orawsp:contentType="optional" orawsp:name="csf-key"> <orawsp:Value/></pre>
csf.map	<p>Oracle WSM map in the credential store that contains the CSF aliases.</p> <p>Default setting:</p> <pre><orawsp:Property orawsp:contentType="optional" orawsp:name="csf.map" orawsp:type="string"/></pre> <p>You can override the default, domain-level Oracle WSM map, by specifying an application-level map name as a <code>Value</code> in this property. For example:</p> <pre><orawsp:Property orawsp:contentType="optional" orawsp:name="csf.map" orawsp:type="string"/> <orawsp:Value>app-level-mapname.map</orawsp:Value> </orawsp:Property></pre> <p>Accessing an application-level map also requires granting credential access and identity permission to the <code>wsm-agent-core.jar</code>.</p>

Table 18-28 (Cont.) http_oauth2_token_client_template Configuration Properties

Name	Description
federated.client.token	<p>Optional property which, by default, specifies that a JWT token is generated for the client using the values of the <code>oauth2.client.csf.key</code> and <code>keystore.sig.csf.key</code> properties.</p> <p>If set to <code>false</code>, <code>oauth2.client.csf.key</code> is used to generate an Authorization header sent in the client request to the OAuth server.</p> <p>Default setting:</p> <pre data-bbox="618 527 1208 659"><orawsp:Property orawsp:contentType="optional" orawsp:name="federated.client.token" orawsp:type="boolean"> <orawsp:Value/> <orawsp:DefaultValue>true</orawsp:DefaultValue></pre>
include.certificate	<p>When <code>true</code>, 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.</p> <p>When <code>false</code>, only the thumbprint and alias of the certificate are included in the JWT token.</p> <p>Default setting:</p> <pre data-bbox="618 940 1219 1100"><orawsp:Property orawsp:contentType="optional" orawsp:name="include.certificate" orawsp:type="string"> <orawsp:Value/> <orawsp:DefaultValue>false</orawsp:DefaultValue> </orawsp:Property></pre>
issuer.name	<p>Optional property that specifies the issuer name used for the locally-generated JWT token (<code>iss:claim</code>). By default it is <code>www.oracle.com</code>.</p> <p>Default setting:</p> <pre data-bbox="618 1255 1333 1394"><orawsp:Property orawsp:contentType="optional" orawsp:name="issuer.name" orawsp:type="string"> <orawsp:Value/> <orawsp:DefaultValue>www.oracle.com</orawsp:DefaultValue></pre>
keystore.sig.csf.key	<p>Optional property that specifies the tenant key from the Oracle WSM keystore for signing the locally-created JWT token.</p> <p>Default setting:</p> <pre data-bbox="618 1549 1195 1656"><orawsp:Property orawsp:contentType="optional" orawsp:name="keystore.sig.csf.key" orawsp:type="string"> <orawsp:Value/></pre>

Table 18-28 (Cont.) http_oauth2_token_client_template Configuration Properties

Name	Description
oauth2.client.csf.key	<p>Required property that specifies the key to use to obtain the client username and password.</p> <p>The value of <code>oauth2.client.csf.key</code> must match the client ID and secret expected by the client profile, as described in "Understanding OAuth Client Profiles Configuration" in <i>Administrator's Guide for Oracle Access Manager with Oracle Security Token Service</i>.</p> <p>If <code>federated.client.token</code> is set to <code>false</code>, <code>oauth2.client.csf.key</code> is used to generate an Authorization header sent in the client request to the OAuth server. If you override <code>oauth2.client.csf.key</code>, that value is used. Otherwise, the value of <code>oauth2.client.csf.key</code> in <code>oauth2_config_client_policy</code> is used.</p> <p>Default setting:</p> <pre data-bbox="618 682 1209 848"><orawsp:Property orawsp:type="string" orawsp:contentType="required" orawsp:name="oauth2.client.csf.key"> <orawsp:Value/> <orawsp:DefaultValue>NONE</orawsp:DefaultValue> </orawsp:Property></pre>
oracle.oauth2.service	<p>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 <code>issuer.name</code> is ignored.</p> <p>When false, the issuer is determined by <code>issuer.name</code> with the default value of <code>"www.oracle.com"</code>.</p>
propagate.identity.context	<p>Optional property that specifies whether the identity context information is propagated as claims in the JWT token.</p> <p>Default setting:</p> <pre data-bbox="618 1203 1195 1310"><orawsp:Property orawsp:contentType="optional" orawsp:name="propagate.identity.context" orawsp:type="string"> <orawsp:Value/></pre>
redirect.uri	<p>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.</p> <p>Default setting:</p> <pre data-bbox="618 1463 1195 1570"><orawsp:Property orawsp:contentType="optional" orawsp:name="redirect.uri" orawsp:type="string"> <orawsp:Value/></pre>

Table 18-28 (Cont.) http_oauth2_token_client_template Configuration Properties

Name	Description
reference.priority	<p>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.</p> <p>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.</p> <p>For more information, see "Specifying the Priority of a Policy Attachment".</p> <p>Default setting:</p> <pre data-bbox="618 674 1195 751"><orawsp:Property orawsp:contentType="optional" orawsp:name="reference.priority" orawsp:type="string"/></pre>
scope	<p>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.</p> <p>Default setting:</p> <pre data-bbox="618 905 1195 982"><orawsp:Property orawsp:contentType="optional" orawsp:name="scope" orawsp:type="string"> <orawsp:Value/></pre> <p>The scope depends on the value of the <code>oracle.oauth2.service</code> property:</p> <ul data-bbox="618 1060 1458 1207" style="list-style-type: none"> • If <code>oracle.oauth2.service</code> is false (the default), the <code>scope</code> property determines the scope. • If <code>oracle.oauth2.service</code> is true and <code>scope</code> has no value, (the default), the protocol, host and port (if available) are obtained from the service URL and used.
subject.precedence	<p>Property that specifies the location from which the subject used to create the JWT token should be obtained.</p> <p>As described in Table 10-2:</p> <ul data-bbox="618 1325 1458 1444" style="list-style-type: none"> • If <code>subject.precedence</code> is set to true, the user name to create the JWT token is obtained only from the authenticated subject. • If <code>subject.precedence</code> is set to false, the user name to create the JWT token is obtained only from the <code>csf-key</code> property. <p>Default setting:</p> <pre data-bbox="618 1503 1325 1612"><orawsp:Property orawsp:contentType="optional" orawsp:name="subject.precedence" orawsp:type="string"> <orawsp:Value>true</orawsp:Value> </orawsp:Property></pre>

Table 18-28 (Cont.) http_oauth2_token_client_template Configuration Properties

Name	Description
time.in.millis	<p>Support standard NumericDate (seconds after Epoch as unit for values in exp (Expiry) and iat (Issued AT) claims in JWT token.</p> <p>If true, then milliseconds after Epoch is used. Otherwise, seconds after Epoch is used.</p> <p>Default setting:</p> <pre data-bbox="618 495 1211 659"><orawsp:Property orawsp:type="boolean" orawsp:contentType="optional" orawsp:name="time.in.millis"> <orawsp:Value/> <orawsp:DefaultValue>true</orawsp:DefaultValue> </orawsp:Property></pre>
user.attributes	<p>Optional property that specifies whether user attributes are inserted as claims in JWT token.</p> <p>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.</p> <p>Requires that the Subject is available and <code>subject.precedence</code> is set to true.</p> <p>A client policy reads the values of the attributes specified using <code>user.attributes</code> from the configured identity store. All valid attribute names and values are used to create JWT claims.</p> <p>The <code>user.attributes</code> 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".</p> <p>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" for more information.</p> <p>Default setting:</p> <pre data-bbox="618 1348 1195 1457"><orawsp:Property orawsp:contentType="optional" orawsp:name="user.attributes" orawsp:type="string"> <orawsp:Value/></pre>
user.roles.include	<p>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.</p> <p>Default setting:</p> <pre data-bbox="618 1642 1224 1772"><orawsp:Property orawsp:contentType="optional" orawsp:name="user.roles.include" orawsp:type="boolean"> <orawsp:Value/> <orawsp:DefaultValue>>false</orawsp:DefaultValue></pre>
user.tenant.name	Reserved for internal use.

Table 18-28 (Cont.) http_oauth2_token_client_template Configuration Properties

Name	Description
set.client.id	<p>Set.client.id is set to false by default. If it is set to true OWSM sends client id to OAuth2 provider in access token request as query param. Default setting:</p> <pre><orawsp:Property orawsp:type="boolean" orawsp:contentType="optional" orawsp:name="set.client.id"> <orawsp:Value/> <orawsp:DefaultValue>false</orawsp:DefaultValue></pre>

18.25 oracle/http_jwt_token_service_template

The 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 18-37](#) for information about the settings.

Configuration

[Table 18-29](#) lists the configuration properties and the default settings for the http_jwt_token_service_template assertion template.

Table 18-29 http_jwt_token_service_template Configuration Properties

Name	Default Values
trusted.issuers	<p>A comma-separated list of trusted issuers for an application that will override the trusted issuers defined at the domain level.</p> <p>Default setting:</p> <pre><orawsp:Property orawsp:contentType="optional" orawsp:name="saml.trusted.issuers" orawsp:type="string"> <orawsp:Value/> </orawsp:Property></pre>
csf.map	<p>Oracle WSM map in the credential store that contains the CSF aliases.</p> <p>Default setting:</p> <pre><orawsp:Property orawsp:contentType="optional" orawsp:name="csf.map" orawsp:type="string"/></pre>
keystore.sig.csf.key	<p>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.</p> <p>Default setting:</p> <pre><orawsp:Property orawsp:contentType="optional" orawsp:name="keystore.sig.csf.key" orawsp:type="string"/></pre>

Table 18-29 (Cont.) http_jwt_token_service_template Configuration Properties

Name	Default Values
reference.priority	<p>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.</p> <p>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.</p> <p>For more information, see "Specifying the Priority of a Policy Attachment".</p> <p>Default setting:</p> <pre data-bbox="581 674 1295 730"><orawsp:Property orawsp:contentType="optional" orawsp:name="reference.priority" orawsp:type="string"/></pre>
propagate.identity.context	<p>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.</p> <p>Default setting:</p> <pre data-bbox="581 911 1154 989"><orawsp:Property orawsp:contentType="optional" orawsp:name="propagate.identity.context" orawsp:type="string"><orawsp:Value/></pre>

18.26 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 18-30](#) lists the settings for the http_oauth2_token_over_ssl_client_template assertion template.

Table 18-30 http_oauth2_token_over_ssl_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	<p>Authentication mechanism.</p> <p>Valid values include:</p> <ul style="list-style-type: none"> • basic—Client authenticates itself by transmitting the username and password. <p>Note: It is recommended that you configure SSL when using basic authentication.</p> <ul style="list-style-type: none"> • 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. 	<pre><orasp:auth-header orasp:mechanism="oauth2"/></pre>
Authentication Header—Header Name	Name of the authentication header.	None
Authentication Header— <i>is-signed</i>	Flag that specifies whether the token is signed.	<pre><orasp:auth-header orasp:is-signed="false"/></pre>
Authentication Header— <i>is encrypted</i>	Flag that specifies whether the token is encrypted.	<pre><orasp:auth-header orasp:is-encrypted="false"/></pre>
Transport Security	Flag that specifies whether SSL is enabled.	<pre><orasp:auth-header orasp:require-tls/></pre>

Table 18-30 (Cont.) http_oauth2_token_over_ssl_client_template Settings

Name	Description	Default Value
Transport Security—Mutual Authentication Required	Flag that specifies whether two-way authentication is required. Valid values include: <ul style="list-style-type: none"> • 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. 	<pre><orasp:auth-header orasp:mutual-auth="false"/></pre>
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.	<pre><orasp:auth-header orasp:include-timestamp="false"/></pre>

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 18-27](#) for information about the settings.

18.27 oracle/http_mutual_auth_over_ssl_client_template

This topic describes the `http_mutual_auth_over_ssl_client_template` assertion template.

Display Name: http mutual auth over ssl client template

Category: Security

Type: http-security

Description

The `http_mutual_auth_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. 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.

Settings

`wss_http_token_over_ssl_client_template` Settings lists the settings for the `http_mutual_auth_over_ssl_client_template` assertion template.

Table 18-31 http_mutual_auth_over_ssl_client_template Settings

Name	Default Value
Authentication Header	
Authentication Header—Mechanism	basic
Transport Layer Security	
Transport Layer Security	Enabled
Transport Layer Security—Mutual Authentication Required	Enabled
Transport Layer Security—Include Timestamp	Disabled
Algorithm Suite	BASIC_128

Configuration

wss_http_token_over_ssl_client_template Configuration Properties lists the configuration properties and the default settings for the wss_http_token_over_ssl_client_template assertion template.

Table 18-32 wss_http_token_over_ssl_client_template Configuration Properties

Name	Default Value	Type
csf-key	basic.credentials	Required
role	ultimateReceiver	Constant
reference.priority	None	Optional

 **See Also:**

- [Table 18-51](#)
- [Authentication Header—Mechanism](#)
- [Transport Layer Security](#)
- [Transport Layer Security—Mutual Authentication Required](#)
- [Transport Layer Security—Include Timestamp](#)
- [Algorithm Suite](#)
- [Table 18-52](#)
- [csf-key](#)
- [role](#)
- [reference.priority](#)

18.28 oracle/http_mutual_auth_over_ssl_service_template

Display Name: http mutual auth over ssl service template

Category: Security

Type: http-security

Description

The `http_mutual_auth_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 `http_mutual_auth_over_ssl_service_template` assertion template are identical to the client version of the assertion template.

Configuration

`wss_http_token_over_ssl_service_template` Configuration Properties lists the configuration properties and the default settings for the `http_mutual_auth_over_ssl_service_template` assertion template.

Table 18-33 `http_mutual_auth_over_ssl_service_template` Configuration Properties

Name	Default Value	Type
realm	owsm	Constant
role	ultimateReceiver	Constant
reference.priority	None	Optional

See Also:

- [oracle/http_mutual_auth_over_ssl_client_template](#)
- [Table 18-53](#)
- [realm](#)
- [role](#)
- [reference.priority](#)

18.29 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 18-39](#) for information about the settings.

Configuration

[Table 18-34](#) lists the configuration properties and the default settings for the `http_jwt_token_over_ssl_service_template` assertion template.

Table 18-34 http_jwt_token_over_ssl_service_template Configuration Properties

Name	Default Values
csf.map	<p>Oracle WSM map in the credential store that contains the CSF aliases.</p> <p>Default setting:</p> <pre data-bbox="581 401 1154 453"><orawsp:Property orawsp:contentType="optional" orawsp:name="csf.map" orawsp:type="string"/></pre>
keystore.sig.csf.key	<p>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.</p> <p>Default setting:</p> <pre data-bbox="581 663 1321 716"><orawsp:Property orawsp:contentType="optional" orawsp:name="keystore.sig.csf.key" orawsp:type="string"/></pre>
propagate.identity.context	<p>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.</p> <p>Default setting:</p> <pre data-bbox="581 905 1154 978"><orawsp:Property orawsp:contentType="optional" orawsp:name="propagate.identity.context" orawsp:type="string"><orawsp:Value/></pre>
reference.priority	<p>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.</p> <p>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.</p> <p>For more information, see "Specifying the Priority of a Policy Attachment".</p> <p>Default setting:</p> <pre data-bbox="581 1377 1295 1430"><orawsp:Property orawsp:contentType="optional" orawsp:name="reference.priority" orawsp:type="string"/></pre>
trusted.issuers	<p>A comma-separated list of trusted issuers for an application that will override the trusted issuers defined at the domain level.</p> <p>Default setting:</p> <pre data-bbox="581 1587 1308 1692"><orawsp:Property orawsp:contentType="optional" orawsp:name="saml.trusted.issuers" orawsp:type="string"> <orawsp:Value/> </orawsp:Property></pre>

18.30 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 18-35](#) lists the settings for the `oauth2_config_client_template` assertion template.

Table 18-35 `oauth2_config_client_template` Settings

Name	Description	Default Value
token-uri	Required property that specifies the token endpoint of the OAuth2 server.	<code>orasp:token-uri="http://host:port/tokens"</code>

Configurations

[Table 18-36](#) lists the default configuration properties for the `oauth2_config_client_template` assertion template.

Table 18-36 `oauth2_config_client_template` Configuration Properties

Name	Description
<code>oauth2.client.csf.key</code>	<p>Required property that specifies the key to use to obtain the client username and password.</p> <p>The value of <code>oauth2.client.csf.key</code> must match the client ID and secret expected by the client profile, as described in "Understanding OAuth Client Profiles Configuration" in <i>Administrator's Guide for Oracle Access Manager with Oracle Security Token Service</i>.</p> <p>Default setting:</p> <pre><orasp:Property orasp:type="string" orasp:contentType="required" orasp:name="oauth2.client.csf.key"> <orasp:Value/> <orasp:DefaultValue>basic.client.credentials</ orasp:DefaultValue> </orasp:Property></pre>
<code>role</code>	<p>SOAP role.</p> <p>Default setting:</p> <pre><orasp:Property orasp:contentType="constant" orasp:name="role" orasp:type="string"> <orasp:DefaultValue> ultimateReceiver </orasp:DefaultValue> </orasp:Property></pre>

Table 18-36 (Cont.) oauth2_config_client_template Configuration Properties

Name	Description
reference.priority	<p>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.</p> <p>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.</p> <p>For more information, see "Specifying the Priority of a Policy Attachment".</p> <p>Default setting:</p> <pre data-bbox="672 674 1377 724"><orawsp:Property orawsp:contentType="optional" orawsp:name="reference.priority" orawsp:type="string"/></pre>
token.uri	<p>Optional property to override the token-uri value.</p> <p>Default setting:</p> <pre data-bbox="672 852 1414 961"><orawsp:Property orawsp:contentType="optional" orawsp:name="token.uri" orawsp:type="string"><orawsp:Value/ ><orawsp:DefaultValue>http://host:port/tokens </orawsp:DefaultValue></orawsp:Property></pre>

18.31 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 18-37](#) lists the settings for the http_jwt_token_client_template assertion template.

Table 18-37 http_jwt_token_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	<p>Authentication mechanism.</p> <p>Valid values include:</p> <ul style="list-style-type: none"> • basic—Client authenticates itself by transmitting the username and password. <p>Note: It is recommended that you configure SSL when using basic authentication.</p> <ul style="list-style-type: none"> • 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. 	<pre><orasp:auth-header orasp:mechanism="jwt"/></pre>
Authentication Header—Header Name	Name of the authentication header.	None
Authentication Header—algorithm-suite	Algorithm suite used to sign the JWT token.	<pre><orasp:auth-header orasp:algorithm- suite="Basic256Sha256"/></pre>
Authentication Header— <i>is-signed</i>	Flag that specifies whether the JWT token is signed. The only valid value for JWT policies is: <i>true</i> .	<pre><orasp:auth-header orasp:is-signed="true"/></pre>
Authentication Header— <i>is encrypted</i>	Flag that specifies whether the JWT token is encrypted.	<pre><orasp:auth-header orasp:is-encrypted="false"/></pre>

Configuration

[Table 18-38](#) lists the configuration properties and the default settings for the `http_jwt_token_client_template` assertion template.

Table 18-38 http_jwt_token_client_template Configuration Properties

Name	Default Values
audience.uri	<p>Audience restriction. The following conditions are supported:</p> <ul style="list-style-type: none"> • If this property is not set, the service URL is used as the audience URI • If this property is set to <code>NONE</code> (not case sensitive), then the audience URI is set to null. • If this property is set to a value other than <code>NONE</code>, then the audience URI is set to this value. <p>Default setting:</p> <pre data-bbox="581 562 1208 667"><orawsp:Property orawsp:contentType="optional" orawsp:name="audience.uri" orawsp:type="string"> <orawsp:Value/> </orawsp:Property></pre>
csf-key	<p>Credential Store Key that maps to a username and password in the Oracle Platform Security Services (OPSS) identity store.</p> <p>Default setting:</p> <pre data-bbox="581 827 1195 932"><orawsp:Property orawsp:contentType="optional" orawsp:name="csf-key" orawsp:type="string"> <orawsp:Value>basic.credentials</orawsp:Value> </orawsp:Property></pre>
csf.map	<p>Oracle WSM map in the credential store that contains the CSF aliases.</p> <p>Default setting:</p> <pre data-bbox="581 1058 1156 1108"><orawsp:Property orawsp:contentType="optional" orawsp:name="csf.map" orawsp:type="string"/></pre>
issuer.name	<p>Name of the JWT issuer. The default value is <code>www.oracle.com</code>.</p> <p>Default setting:</p> <pre data-bbox="581 1234 1195 1339"><orawsp:Property orawsp:contentType="optional" orawsp:name="issuer.name" orawsp:type="string"> <orawsp:Value>www.oracle.com</orawsp:Value> </orawsp:Property></pre>
keystore.sig.csf.key	<p>The alias and password used for storing the signature key password in the keystore. If specified, the key corresponding to this <code>csf-key</code> 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.</p> <p>Default setting:</p> <pre data-bbox="581 1554 1321 1604"><orawsp:Property orawsp:contentType="optional" orawsp:name="keystore.sig.csf.key" orawsp:type="string"/></pre>
propagate.identity.context	<p>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.</p> <p>Default setting:</p> <pre data-bbox="581 1785 1156 1869"><orawsp:Property orawsp:contentType="optional" orawsp:name="propagate.identity.context" orawsp:type="string"><orawsp:Value/></pre>

Table 18-38 (Cont.) http_jwt_token_client_template Configuration Properties

Name	Default Values
reference.priority	<p>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.</p> <p>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.</p> <p>For more information, see "Specifying the Priority of a Policy Attachment".</p> <p>Default setting:</p> <pre data-bbox="581 674 1295 724"><orawsp:Property orawsp:contentType="optional" orawsp:name="reference.priority" orawsp:type="string"/></pre>
subject.precedence	<p>Property that specifies the location from which the subject used to create the JWT token should be obtained.</p> <p>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.</p> <p>Default setting:</p> <pre data-bbox="581 1010 1284 1119"><orawsp:Property orawsp:contentType="optional" orawsp:name="subject.precedence" orawsp:type="string"> <orawsp:Value>true</orawsp:Value> </orawsp:Property></pre>
user.attributes	<p>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.</p> <p>Requires that the Subject is available and subject.precedence is set to true.</p> <p>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.</p> <p>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".</p> <p>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" for more information.</p> <p>Default setting:</p> <pre data-bbox="581 1776 1247 1829"><orawsp:Property orawsp:contentType="optional" orawsp:name="user.attributes" orawsp:type="string"/></pre>

Table 18-38 (Cont.) http_jwt_token_client_template Configuration Properties

Name	Default Values
user.roles.include	User roles to be included in the JWT token. If set to <code>true</code> , the authenticated user roles are included in the JWT token as private claims. The default is <code>false</code> . Default setting: <pre><orawsp:Property orawsp:contentType="optional" orawsp:name="user.roles.include" orawsp:type="string"> <orawsp:Value>false</orawsp:Value> </orawsp:Property></pre>
user.tenant.name	Reserved for internal use.

18.32 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 18-39](#) lists the settings for the `http_jwt_token_over_ssl_client_template` assertion template.

Table 18-39 http_jwt_token_over_ssl_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	<p>Authentication mechanism.</p> <p>Valid values include:</p> <ul style="list-style-type: none"> • basic—Client authenticates itself by transmitting the username and password. <p>Note: It is recommended that you configure SSL when using basic authentication.</p> <ul style="list-style-type: none"> • 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. 	<pre><orasp:auth-header orasp:mechanism="jwt"/></pre>
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.	<pre><orasp:auth-header orasp:algorithm- suite="Basic256Sha256"/></pre>
Authentication Header— <i>is-signed</i>	Flag that specifies whether the JWT token is signed. The only valid value for JWT policies is: <i>true</i> .	<pre><orasp:auth-header orasp:is-signed="true"/></pre>
Authentication Header— <i>is encrypted</i>	Flag that specifies whether the JWT token is encrypted.	<pre><orasp:auth-header orasp:is-encrypted="false"/></pre>
Transport Security	Flag that specifies whether SSL is enabled.	<pre><orasp:auth-header orasp:require-tls/></pre>

Table 18-39 (Cont.) http_jwt_token_over_ssl_client_template Settings

Name	Description	Default Value
Transport Security—Mutual Authentication Required	<p>Flag that specifies whether two-way authentication is required.</p> <p>Valid values include:</p> <ul style="list-style-type: none"> 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. 	<pre><orasp:auth-header orasp:mutual-auth="false"/></pre>
Transport Security—Include Timestamp	<p>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.</p>	<pre><orasp:auth-header orasp:include-timestamp="false"/></pre>

Configuration

Table 18-40 lists the configuration properties and the default settings for the http_jwt_token_over_ssl_client_template assertion template.

Table 18-40 http_jwt_token_over_ssl_client_template Configuration Properties

Name	Default Values
audience.uri	<p>Audience restriction. The following conditions are supported:</p> <ul style="list-style-type: none"> 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. <p>Default setting:</p> <pre><orawsp:Property orawsp:contentType="optional" orawsp:name="audience.uri" orawsp:type="string"> <orawsp:Value/> </orawsp:Property></pre>
csf.map	<p>Oracle WSM map in the credential store that contains the CSF aliases.</p> <p>Default setting:</p> <pre><orawsp:Property orawsp:contentType="optional" orawsp:name="csf.map" orawsp:type="string"/></pre>

Table 18-40 (Cont.) http_jwt_token_over_ssl_client_template Configuration Properties

Name	Default Values
csf-key	<p>Credential Store Key that maps to a username and password in the Oracle Platform Security Services (OPSS) identity store.</p> <p>Default setting:</p> <pre data-bbox="581 428 1195 541"><orawsp:Property orawsp:contentType="optional" orawsp:name="csf-key" orawsp:type="string"> <orawsp:Value>basic.credentials</orawsp:Value> </orawsp:Property></pre>
issuer.name	<p>Name of the JWT issuer. The default value is www.oracle.com.</p> <p>Default setting:</p> <pre data-bbox="581 663 1195 772"><orawsp:Property orawsp:contentType="optional" orawsp:name="issuer.name" orawsp:type="string"> <orawsp:Value>www.oracle.com</orawsp:Value> </orawsp:Property></pre>
keystore.sig.csf.key	<p>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.</p> <p>Default setting:</p> <pre data-bbox="581 982 1321 1035"><orawsp:Property orawsp:contentType="optional" orawsp:name="keystore.sig.csf.key" orawsp:type="string"/></pre>
propagate.identity.context	<p>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.</p> <p>Default setting:</p> <pre data-bbox="581 1220 1156 1297"><orawsp:Property orawsp:contentType="optional" orawsp:name="propagate.identity.context" orawsp:type="string"><orawsp:Value/></pre>
reference.priority	<p>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.</p> <p>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.</p> <p>For more information, see "Specifying the Priority of a Policy Attachment".</p> <p>Default setting:</p> <pre data-bbox="581 1696 1295 1749"><orawsp:Property orawsp:contentType="optional" orawsp:name="reference.priority" orawsp:type="string"/></pre>

Table 18-40 (Cont.) http_jwt_token_over_ssl_client_template Configuration Properties

Name	Default Values
subject.precedence	<p>Property that specifies the location from which the subject used to create the JWT token should be obtained.</p> <p>If <code>subject.precedence</code> is set to <code>true</code>, the user name to create the JWT token is obtained only from the authenticated Subject. If <code>subject.precedence</code> is set to <code>false</code>, the user name to create the JWT token is obtained only from the <code>csf-key</code> username property.</p> <p>Default setting:</p> <pre data-bbox="581 558 1284 667"><orawsp:Property orawsp:contentType="optional" orawsp:name="subject.precedence" orawsp:type="string"> <orawsp:Value>true</orawsp:Value> </orawsp:Property></pre>
user.attributes	<p>List of user attributes for the authenticated user to be included in the JWT token.</p> <p>Specify the attributes to be included as a comma-separated list. For example, <code>attrib1,attrib2</code>. 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.</p> <p>Requires that the Subject is available and <code>subject.precedence</code> is set to <code>true</code>.</p> <p>A client policy reads the values of the attributes specified using <code>user.attributes</code> from the configured identity store. All valid attribute names and values are used to create JWT claims.</p> <p>The <code>user.attributes</code> 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".</p> <p>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" for more information.</p> <p>Default setting:</p> <pre data-bbox="581 1325 1247 1375"><orawsp:Property orawsp:contentType="optional" orawsp:name="user.attributes" orawsp:type="string"/></pre>
user.roles.include	<p>User roles to be included in the JWT token. If set to <code>true</code>, the authenticated user roles are included in the JWT token as private claims. The default is <code>false</code>.</p> <p>Default setting:</p> <pre data-bbox="581 1535 1284 1642"><orawsp:Property orawsp:contentType="optional" orawsp:name="user.roles.include" orawsp:type="string"> <orawsp:Value>false</orawsp:Value> </orawsp:Property></pre>
user.tenant.name	Reserved for use internal use.

18.33 oracle/wss10_message_protection_client_template

This topic describes the `wss10_message_protection_client_template` assertion template.

Display Name: Wss10 Message Protection client Assertion Template

Category: Security

Type: wss10-anonymous-with-certificates

Description

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 18-41](#) lists the settings for the `wss10_message_protection_client_template` assertion template.

Table 18-41 wss10_message_protection_client_template Settings

Name	Default Value
X509 Token	
Sign Key Reference Mechanism	direct
Encryption Key Reference Mechanism	direct
Recipient Sign Key Reference Mechanism	direct
Recipient Encryption Key Reference Mechanism	direct
Is Signed	Disabled
Use PKI Path	Disabled
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation versions 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Enabled
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Disabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

Table 18-42 lists the configuration properties and the default settings for the `wss10_message_protection_client_template` assertion template.

Table 18-42 `wss10_message_protection_client_template` Configuration Properties

Name	Default Value	Type
<code>keystore.recipient.alias</code>	<code>orakey</code>	Required
<code>role</code>	<code>ultimateReceiver</code>	Constant
<code>keystore.sig.csf.key</code>	None	Optional
<code>keystore.enc.csf.key</code>	None	Optional
<code>ignore.timestamp.in.response</code>	<code>false</code>	Optional
<code>sc.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.34 oracle/wss10_message_protection_service_template

This topic describes the `wss10_message_protection_service_template` assertion template.

Display Name: Wss10 Message Protection service Assertion Template

Category: Security

Type: wss10-anonymous-with-certificates

Description

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 18-41 for information about the settings.

Configuration

Table 18-43 lists the configuration properties and the default settings for the `wss10_message_protection_service_template` assertion template.

Table 18-43 `wss10_message_protection_service_template` Configuration Properties

Name	Default Value	Type
<code>role</code>	<code>ultimateReceiver</code>	Constant
<code>keystore.sig.csf.key</code>	None	Optional
<code>keystore.enc.csf.key</code>	None	Optional
<code>sc.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.35 oracle/wss11_message_protection_client_template

This topic describes the `wss11_message_protection_client_template` assertion template.

Display Name: Wss11 Message Protection client Assertion Template

Category: Security

Type: wss11-anonymous-with-certificates

Description

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 18-44](#) lists the settings for the `wss11_message_protection_client_template` assertion template.

Table 18-44 wss11_message_protection_client_template Settings

Name	Default Value
X509 Token	
Encryption Key Reference Mechanism	thumbprint
Is Signed	Enabled
Use PKI Path	Disabled
Derived Keys	Disabled
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Enabled
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Enabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

Table 18-45 lists the configuration properties and the default settings for the `wss11_message_protection_client_template` assertion template.

Table 18-45 `wss11_message_protection_client_template` Configuration Properties

Name	Default Value	Type
<code>keystore.recipient.alias</code>	<code>orakey</code>	Required
<code>role</code>	<code>ultimateReceiver</code>	Constant
<code>keystore.enc.csf.key</code>	None	Optional
<code>ignore.timestamp.in.response</code>	<code>false</code>	Optional
<code>sc.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.36 oracle/wss11_message_protection_service_template

This topic describes the `wss11_message_protection_service_template` assertion template.

Display Name: Wss11 Message Protection service Assertion Template

Category: Security

Type: `wss11-anonymous-with-certificates`

Description

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 18-44 for information about the settings.

Configuration

Table 18-46 lists the configuration properties and the default settings for the `wss11_message_protection_service_template` assertion template.

Table 18-46 `wss11_message_protection_service_template` Configuration Properties

Name	Default Value	Type
<code>role</code>	<code>ultimateReceiver</code>	Constant
<code>keystore.enc.csf.key</code>	None	Optional
<code>sc.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.37

wss11_username_token_derivedkey_message_protection_signature_client

This topic describes the `oracle/wss11_username_token_derivedkey_with_message_protection_signature_only_client_template` assertion template.

Display Name: wss11 username with derivedKey with message protection signature only client template

Category: Security

Type: wss11-username-with-derivedKey

 **Note:**

When cloning `wss11-username-with-derivedKey` assertion based policies, the request, response or fault Message part can either contain signed parts or encrypted parts . Both are not supported.

Description

The `wss11_username_token_derivedkey_with_message_protection_signature_only_client_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 the outgoing SOAP message. The web service provider 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.

Settings

[Table 18-100](#) lists the settings for the `wss11_username_token_derivedkey_with_message_protection_signature_only_client_template` assertion template.

Table 18-47 `wss11_username_token_derivedkey_with_message_protection_signature_only_client_template`

Name	Default Value
Username Token	
Password Type	none
Creation Time Required	Disabled
Nonce Required	Disabled
Is Encrypted	Disables
Is Signed	Enabled

Table 18-47 (Cont.)

wss11_username_token_derivedkey_with_message_protection_signature_only_client_template

Name	Default Value
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Enabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

Table 18-101 lists the configuration properties and the default settings for the assertion template.

Table 18-48 wss11_username_token_derivedkey_with_message_protection_signature_only_client_template Configuration Properties

Name	Default Value	Type
csf-key	basic.credentials	Required
user.tenant.name	None	Optional
ignore.timestamp.in.response	false	Optional
iterations	1000	Optional

18.38

wss11_username_token_derivedkey_message_protection_encryption_client_template

This topic describes the oracle/wss11_username_token_derivedkey_with_message_protection_encryption_only_client_template assertion template.

Display Name: wss11 username token derivedKey with message protection encryption only client template

Category: Security

Type: wss11-username-with-derivedKey



Note:

When cloning `wss11-username-with-derivedKey` assertion based policies, the request, response or fault Message part can either contain signed parts or encrypted parts . Both are not supported.

Description

The `wss11_username_token_derivedkey_with_message_protection_encryption_only_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 encrypts the outgoing SOAP message.

The web service provider 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.

Settings

[Table 18-100](#) lists the settings for the `wss11_username_token_derivedkey_with_message_protection_encryption_only_client_template` assertion template.

Table 18-49 `wss11_username_token_derivedkey_with_message_protection_encryption_only_client_template`

Name	Default Value
Username Token	
Password Type	none
Creation Time Required	Disabled
Nonce Required	Disabled
Is Encrypted	Disabled
Is Signed	Disabled
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Disabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

[Table 18-101](#) lists the configuration properties and the default settings for the assertion template.

Table 18-50 wss11_username_token_derivedkey_with_message_protection_encryption_only_client_template Configuration Properties

Name	Default Value	Type
csf-key	basic.credentials	Required
user.tenant.name	None	Optional
ignore.timestamp.in.response	false	Optional
iterations	1000	Optional

18.39 oracle/wss_http_token_over_ssl_client_template

This topic describes the `wss_http_token_over_ssl_client_template` assertion template.

Display Name: Wss HTTP Token Over SSL client Assertion Template

Category: Security

Type: http-security

Description

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. 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.

Settings

[Table 18-51](#) lists the settings for the `wss_http_token_over_ssl_client_template` assertion template.

Table 18-51 wss_http_token_over_ssl_client_template Settings

Name	Default Value
Authentication Header	
Authentication Header—Mechanism	basic
Authentication Header—Header Name	None
Transport Layer Security	
Transport Layer Security	Enabled
Transport Layer Security—Mutual Authentication Required	Disabled
Transport Layer Security—Include Timestamp	Disabled
Algorithm Suite	BASIC_128

Configuration

[Table 18-52](#) lists the configuration properties and the default settings for the `wss_http_token_over_ssl_client_template` assertion template.

Table 18-52 wss_http_token_over_ssl_client_template Configuration Properties

Name	Default Value	Type
csf-key	basic.credentials	Required
role	ultimateReceiver	Constant
reference.priority	None	Optional

18.40 oracle/wss_http_token_over_ssl_service_template

Display Name: Wss HTTP Token Over SSL service Assertion Template

Category: Security

Type: http-security

Description

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 18-51](#) for information about the settings.

Configuration

[Table 18-53](#) lists the configuration properties and the default settings for the `wss_http_token_service_template` assertion template.

Table 18-53 wss_http_token_over_ssl_service_template Configuration Properties

Name	Default Value	Type
realm	owsm	Constant
role	ultimateReceiver	Constant
reference.priority	None	Optional

18.41 oracle/wss_saml_token_bearer_client_template

This topic describes the `wss_saml_token_bearer_client_template` assertion template.

Display Name: Wss SAML Bearer Token client Assertion Template

Category: Security

Type: wss11-saml-token

Description

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 18-54](#) lists the settings for the `wss_saml_token_bearer_client_template` assertion template.

Table 18-54 `wss_saml_token_bearer_client_template` Settings

Name	Default Value
SAML Token Type	
Version	1.1
Confirmation Type	bearer
Name Identifier Format	unspecified

Configuration

[Table 18-55](#) lists the configuration properties and the default settings for the `wss_saml_token_bearer_client_template` assertion template.

Table 18-55 `wss_saml_token_bearer_client_template` Configuration Properties

Name	Default Value	Type
<code>user.attributes</code>	None	Optional
<code>user.roles.include</code>	false	Optional
<code>saml.issuer.name</code>	www.oracle.com	Optional
<code>csf-key</code>	basic.credentials	Optional
<code>csf.map</code>	None	Optional
<code>subject.precedence</code>	true	Optional
<code>saml.audience.uri</code>	None	Optional
<code>keystore.sig.csf.key</code>	None	Optional
<code>saml.envelope.signature.required</code>	true	Optional
<code>propagate.identity.context</code>	None	Optional
<code>user.tenant.name</code>	None	Optional
<code>reference.priority</code>	None	Optional
<code>include-timestamp</code>	false	Optional

18.42 oracle/wss_saml_token_bearer_service_template

This topic describes the `wss_saml_token_bearer_service_template` assertion template.

Display Name: Wss SAML Bearer Token service Assertion Template

Category: Security

Type: wss11-saml-token

Description

The `wss_saml_token_bearer_service_template` assertion template includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method [*Bearer*] is created automatically.

Settings

[Table 18-54](#) lists the settings for the `wss_saml_token_bearer_service_template` assertion template.

Table 18-56 `wss_saml_token_bearer_service_template` Settings

Name	Default Value
SAML Token Type	
Version	1.1
Confirmation Type	bearer
Name Identifier Format	unspecified

Configuration

[Table 18-59](#) lists the configuration properties and the default settings for the `wss_saml_token_bearer_service_template` assertion template.

Table 18-57 `wss_saml_token_bearer_service_template` Configuration Properties

Name	Default Value	Type
role	ultimateReceiver	Constant
saml.trusted.issuers	None	Optional
saml.envelope.signature.required	true	Optional
propagate.identity.context	None	Optional
reference.priority	None	Optional

18.43 oracle/wss_saml_token_bearer_over_ssl_client_template

This topic describes the `wss_saml_token_bearer_over_ssl_client` template assertion template.

Display Name: Wss SAML Token (Confirmation method as bearer) Over SSL client Assertion Template

Category: Security

Type: wss-saml-token-bearer-over-ssl

Description

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 18-58 lists the settings for the `wss_saml_token_bearer_over_ssl_client_template` assertion template.

Table 18-58 `wss_saml_token_bearer_over_ssl_client_template` Settings

Name	Default Value
SAML Token Type	
Version	1.1
Confirmation Type	bearer
Is Signed	Disabled
Is Encrypted	Disabled
Name Identifier Format	unspecified
Transport Layer Security	
Transport Layer Security	Enabled
Transport Layer Security—Mutual Authentication Required	Disabled
Transport Layer Security—Include Timestamp	Enabled
Algorithm Suite	None
Algorithm Suite	BASIC_128
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Disabled
Server Entropy	Enabled

Configuration

Table 18-59 lists the configuration properties and the default settings for the `wss_saml_token_bearer_over_ssl_client_template` assertion template.

Table 18-59 `wss_saml_token_bearer_over_ssl_client_template` Configuration Properties

Name	Default Value	Type
<code>user.attributes</code>	None	Optional
<code>user.roles.include</code>	false	Optional

Table 18-59 (Cont.) wss_saml_token_bearer_over_ssl_client_template Configuration Properties

Name	Default Value	Type
saml.issuer.name	www.oracle.com	Optional
csf-key	basic.credentials	Optional
subject.precedence	true	Optional
saml.audience.uri	None	Optional
keystore.sig.csf.key	None	Optional
propagate.identity.context	None	Optional
user.tenant.name	None	Optional
ignore.timestamp.in.response	false	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.44 oracle/ wss_saml_token_bearer_over_ssl_service_template

This topic describes the `wss_saml_token_bearer_over_ssl_service_template` assertion template.

Display Name: Wss SAML Token (Confirmation method as bearer) Over SSL service Assertion Template

Category: Security

Type: wss-saml-token-bearer-over-ssl

Description

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. See [Table 18-58](#) for information about the settings.

Configuration

[Table 18-60](#) lists the configuration properties and the default settings for the `wss_saml_token_bearer_over_ssl_service_template` assertion template.

Table 18-60 wss_saml_token_bearer_over_ssl_service_template Configuration Properties

Name	Default Value	Type
role	ultimateReceiver	Constant

Table 18-60 (Cont.) wss_saml_token_bearer_over_ssl_service_template Configuration Properties

Name	Default Value	Type
saml.trusted.issuers	None	Optional
propagate.identity.context	None	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.45 oracle/ wss_saml20_token_bearer_over_ssl_client_template

This topic describes the `wss_saml20_token_bearer_over_ssl_client` template assertion template.

Display Name: Wss SAML V2.0 Token (Confirmation method as bearer) Over SSL client Assertion Template

Category: Security

Type: wss-saml-token-bearer-over-ssl

Description

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 18-61](#) lists the settings for the `wss_saml20_token_bearer_over_ssl_client_template` assertion template.

Table 18-61 wss_saml20_token_bearer_over_ssl_client_template Settings

Name	Default Value
SAML Token Type	
Version	2.0
Confirmation Type	bearer
Is Signed	Disabled
Is Encrypted	Disabled
Name Identifier Format	unspecified
Transport Layer Security	
Transport Layer Security	Enabled
Transport Layer Security—Mutual Authentication Required	Disabled
Transport Layer Security—Include Timestamp	Enabled
Algorithm Suite	None
Algorithm Suite	BASIC_128

Table 18-61 (Cont.) wss_saml20_token_bearer_over_ssl_client_template Settings

Name	Default Value
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Disabled
Server Entropy	Enabled

Configuration

Table 18-62 lists the configuration properties and the default settings for the `wss_saml20_token_bearer_over_ssl_client_template` assertion template.

Table 18-62 wss_saml20_token_bearer_over_ssl_client_template Configuration Properties

Name	Default Value	Type
<code>user.attributes</code>	None	Optional
<code>user.roles.include</code>	false	Optional
<code>saml.issuer.name</code>	www.oracle.com	Optional
<code>csf-key</code>	basic.credentials	Optional
<code>subject.precedence</code>	true	Optional
<code>saml.audience.uri</code>	None	Optional
<code>keystore.sig.csf.key</code>	None	Optional
<code>propagate.identity.context</code>	None	Optional
<code>ignore.timestamp.in.response</code>	false	Optional
<code>sc.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.46 oracle/ wss_saml20_token_bearer_over_ssl_service_template

Display Name: Wss SAML V2.0 Token (Confirmation method as bearer) Over SSL service Assertion Template

Category: Security

Type: wss-saml-token-bearer-over-ssl

Description

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. See [Table 18-61](#) for information about the settings.

Configuration

[Table 18-63](#) lists the configuration properties and the default settings for the `wss_saml20_token_bearer_over_ssl_service_template` assertion template.

Table 18-63 `wss_saml20_token_bearer_over_ssl_service_template` Configuration Properties

Name	Default Value	Type
role	<code>ultimateReceiver</code>	Constant
saml.trusted.issuers	None	Optional
propagate.identity.context	None	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.47 oracle/wss_saml_token_over_ssl_client_template

This topic describes the `wss_saml_token_over_ssl_client_template` assertion template.

Display Name: Wss SAML Token Over SSL client Assertion Template

Category: Security

Type: wss-saml-token-over-ssl

Description

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 18-64](#) lists the settings for the `wss_saml_token_over_ssl_client_template` assertion template.

Table 18-64 `wss_saml_token_over_ssl_client_template` Settings

Name	Default Value
SAML Token Type	
Version	1.1

Table 18-64 (Cont.) wss_saml_token_over_ssl_client_template Settings

Name	Default Value
Confirmation Type	sender-vouches
Is Signed	Enabled
Is Encrypted	Disabled
Name Identifier Format	unspecified
Transport Layer Security	
Transport Layer Security	Enabled
Transport Layer Security—Mutual Authentication Required	Enabled
Transport Layer Security—Include Timestamp	Enabled
Algorithm Suite	None
Algorithm Suite	BASIC_128
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Disabled
Server Entropy	Enabled

Configuration

Table 18-65 lists the configuration properties and the default settings for the wss_saml_token_over_ssl_client_template assertion template.

Table 18-65 wss_saml_token_over_ssl_client_template Configuration Properties

Name	Default Value	Type
user.attributes	None	Optional
user.roles.include	false	Optional
saml.issuer.name	www.oracle.com	Optional
csf-key	basic.credentials	Optional
subject.precedence	true	Optional
saml.audience.uri	None	Optional
propagate.identity.context	None	Optional
ignore.timestamp.in.response	false	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.48 oracle/wss_saml_token_over_ssl_service_template

This topic describes the `wss_saml_token_over_ssl_service_template` assertion template.

Display Name: Wss SAML Token Over SSL service Assertion Template

Category: Security

Type: wss-saml-token-over-ssl

Description

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. See [Table 18-64](#) for information about the settings.

Configuration

[Table 18-66](#) lists the configuration properties and the default settings for the `wss_saml_token_over_ssl_service_template` assertion template.

Table 18-66 `wss_saml_token_over_ssl_service_template` Configuration Properties

Name	Default Value	Type
role	<code>ultimateReceiver</code>	Constant
saml.trusted.issuers	None	Optional
propagate.identity.context	None	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.49 oracle/wss_saml20_token_over_ssl_client_template

This topic describes the `wss_saml20_token_over_ssl_client_template` assertion template.

Display Name: Wss SAML V2.0 Token Over SSL client Assertion Template

Category: Security

Type: wss-saml-token-over-ssl

Description

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 18-67 lists the settings for the `wss_saml20_token_over_ssl_client_template` assertion template.

Table 18-67 wss_saml20_token_over_ssl_client_template Settings

Name	Default Value
SAML Token Type	
Version	2.0
Confirmation Type	sender-vouches
Is Signed	Enabled
Is Encrypted	Disabled
Name Identifier Format	unspecified
Transport Layer Security	
Transport Layer Security	Enabled
Transport Layer Security—Mutual Authentication Required	Enabled
Transport Layer Security—Include Timestamp	Enabled
Algorithm Suite	None
Algorithm Suite	BASIC_128
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Disabled
Server Entropy	Enabled

Configuration

Table 18-68 lists the configuration properties and the default settings for the `wss_saml20_token_over_ssl_client_template` assertion template.

Table 18-68 wss_saml20_token_over_ssl_client_template Configuration Properties

Name	Default Value	Type
user.attributes	None	Optional
user.roles.include	false	Optional
saml.issuer.name	www.oracle.com	Optional
csf-key	basic.credentials	Optional
subject.precedence	true	Optional
saml.audience.uri	None	Optional

Table 18-68 (Cont.) wss_saml20_token_over_ssl_client_template Configuration Properties

Name	Default Value	Type
propagate.identity.context	None	Optional
ignore.timestamp.in.response	false	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.50 oracle/wss_saml20_token_over_ssl_service_template

This topic describes the `wss_saml20_token_over_ssl_service_template` assertion template.

Display Name: Wss SAML V2.0 Token Over SSL service Assertion Template

Category: Security

Type: wss-saml-token-over-ssl

Description

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. See [Table 18-67](#) for information about the settings.

Configuration

[Table 18-69](#) lists the configuration properties and the default settings for the `wss_saml20_token_over_ssl_service_template` assertion template.

Table 18-69 wss_saml20_token_over_ssl_service_template Configuration Properties

Name	Default Value	Type
role	ultimateReceiver	Constant
saml.trusted.issuers	None	Optional
propagate.identity.context	None	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.51 oracle/wss_username_token_over_ssl_client_template

This topic describes the `wss_username_token_over_ssl_client_template` assertion template.

Display Name: Wss Username Token Over SSL client Assertion Template

Category: Security

Type: wss-username-token-over-ssl

Description

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.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token.

Settings

Table 18-70 lists the settings for the `wss_username_token_over_ssl_client_template` assertion template.

Table 18-70 `wss_username_token_over_ssl_client_template` Settings

Name	Default Value
Username Token	
Password Type	plaintext
Creation Time Required	Disabled
Nonce Required	Disabled
Transport Layer Security	
Transport Layer Security	Enabled
Transport Layer Security—Mutual Authentication Required	Disabled
Transport Layer Security—Include Timestamp	Enabled
Algorithm Suite	None
Algorithm Suite	BASIC_128
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Disabled
Server Entropy	Enabled

Configuration

Table 18-71 lists the configuration properties and the default settings for the `wss_username_token_over_ssl_client_template` assertion template.

Table 18-71 `wss_username_token_over_ssl_client_template` Configuration Properties

Name	Default Value	Type
role	ultimateReceiver	Constant

Table 18-71 (Cont.) wss_username_token_over_ssl_client_template Configuration Properties

Name	Default Value	Type
csf-key	basic.credentials	Required
csf.map	None	Optional
user.tenant.name	None	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional
ignore.timestamp.in.response	false	Optional

18.52 oracle/wss_username_token_over_ssl_service_template

This topic describes the `wss_username_token_over_ssl_service_template` assertion template.

Display Name: Wss Username Token Over SSL service Assertion Template

Category: Security

Type: wss-username-token-over-ssl

Description

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.

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 18-70](#) for information about the settings.

Configuration

[Table 18-72](#) lists the configuration properties and the default settings for the `wss_username_token_over_ssl_service_template` assertion template.

Table 18-72 wss_username_token_over_ssl_service_template Configuration Properties

Name	Default Value	Type
role	ultimateReceiver	Constant
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.53 oracle/wss10_saml_hok_token_with_message_protection_client_template

This topic describes the `wss10_saml_hok_token_with_message_protection_client_template` assertion template.

Display Name: Wss10 SAML Holder-Of-Key Token with Message Protection client Assertion Template

Category: Security

Type: wss10-saml-hok-with-certificates

Description

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 18-73](#) lists the settings for the `wss10_saml_hok_token_with_message_protection_client_template` assertion template.

Table 18-73 wss10_saml_hok_token_with_message_protection_client_template Settings

Name	Default Value
SAML Token Type	
Version	1.1
Confirmation Type	holder-of-key
Is Signed	Enabled
Is Encrypted	Disabled
Name Identifier Format	unspecified
X509 Token	
Sign Key Reference Mechanism	ski
Encryption Key Reference Mechanism	direct
Recipient Sign Key Reference Mechanism	direct
Recipient Encryption Key Reference Mechanism	direct
Is Signed	Disabled
Use PKI Path	Disabled
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Disabled

Table 18-73 (Cont.) wss10_saml_hok_token_with_message_protection_client_template Settings

Name	Default Value
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

[Table 18-74](#) lists the configuration properties and the default settings for the `wss10_saml_hok_token_with_message_protection_client_template` assertion template.

Table 18-74 wss10_saml_hok_token_with_message_protection_client_template Configuration Properties

Name	Default Value	Type
<code>user.attributes</code>	None	Optional
<code>keystore.recipient.alias</code>	orakey	Required
<code>saml.issuer.name</code>	www.oracle.com	Optional
<code>user.roles.include</code>	false	Optional
<code>saml.assertion.filename</code>	temp	Optional
<code>keystore.sig.csf.key</code>	None	Optional
<code>keystore.enc.csf.key</code>	None	Optional
<code>ignore.timestamp.in.response</code>	false	Optional
<code>reference.priority</code>	None	Optional

18.54 oracle/

wss10_saml_hok_token_with_message_protection_service_template

This topic describes the `wss10_saml_hok_token_with_message_protection_service_template` assertion template

Display Name: Wss10 SAML Holder-Of-Key Token with Message Protection service Assertion Template

Category: Security

Type: wss10-saml-hok-with-certificates

Description

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. See [Table 18-73](#) for information about the settings.

Configuration

[Table 18-75](#) lists the configuration properties and the default settings for the `wss10_saml_hok_token_with_message_protection_service_template` assertion template.

Table 18-75 `wss10_saml_hok_token_with_message_protection_service_template` Configuration Properties

Name	Default Value	Type
<code>role</code>	<code>ultimateReceiver</code>	Constant
<code>keystore.sig.csf.key</code>	None	Optional
<code>keystore.enc.csf.key</code>	None	Optional
<code>saml.trusted.issuers</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.55 oracle/wss10_saml_token_with_message_protection_client_template

This topic describes the `wss10_saml_token_with_message_protection_client_template` assertion template.

Display Name: Wss10 SAML Token with Message Protection client Assertion Template

Category: Security

Type: wss10-saml-with-certificates

Description

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 18-76](#) lists the settings for the `wss10_saml_token_with_message_protection_client_template` assertion template.

Table 18-76 wss10_saml_token_with_message_protection_client_template Settings

Name	Default Value
SAML Token Type	
Version	1.1
Confirmation Type	sender-vouches
Is Signed	Enabled
Is Encrypted	Disabled
Name Identifier Format	unspecified
X509 Token	
Sign Key Reference Mechanism	direct
Encryption Key Reference Mechanism	direct
Recipient Sign Key Reference Mechanism	direct
Recipient Encryption Key Reference Mechanism	direct
Is Signed	Disabled
Use PKI Path	Disabled
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Enabled
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Disabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

[Table 18-77](#) lists the configuration properties and the default settings for the wss10_saml_token_with_message_protection_client_template assertion template.

Table 18-77 wss10_saml_token_with_message_protection_client_template Configuration Properties

Name	Default Value	Type
user.attributes	None	Optional
keystore.recipient.alias	orakey	Required
user.roles.include	false	Optional
saml.issuer.name	www.oracle.com	Optional
keystore.sig.csf.key	None	Optional
keystore.enc.csf.key	None	Optional
csf-key	basic.credentials	Optional
subject.precedence	true	Optional
saml.audience.uri	None	Optional
propagate.identity.context	None	Optional
ignore.timestamp.in.response	false	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.56 oracle/ wss10_saml_token_with_message_protection_service_template

This topic describes the `wss10_saml_token_with_message_protection_service_template` assertion template.

Display Name: Wss10 SAML Token with Message Protection service Assertion Template

Category: Security

Type: wss10-saml-with-certificates

Description

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. See [Table 18-76](#) for information about the settings.

Configuration

Table 18-78 lists the configuration properties and the default settings for the `wss10_saml_token_with_message_protection_service_template` assertion template.

Table 18-78 `wss10_saml_token_with_message_protection_service_template` Configuration Properties

Name	Default Value	Type
<code>role</code>	<code>ultimateReceiver</code>	Constant
<code>keystore.sig.csf.key</code>	None	Optional
<code>keystore.enc.csf.key</code>	None	Optional
<code>saml.trusted.issuers</code>	None	Optional
<code>propagate.identity.context</code>	None	Optional
<code>sc.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.57 oracle/wss10_saml20_token_with_message_protection_client_template

Display Name: Wss10 SAML V2.0 Token with Message Protection client Assertion Template

Category: Security

Type: wss10-saml-with-certificates

Description

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 18-79 lists the settings for the `wss10_saml20_token_with_message_protection_client_template` assertion template.

Table 18-79 `wss10_saml20_token_with_message_protection_client_template` Settings

Name	Default Value
SAML Token Type	
<code>Version</code>	2.0
<code>Confirmation Type</code>	sender-vouches

Table 18-79 (Cont.) wss10_saml20_token_with_message_protection_client_template Settings

Name	Default Value
Is Signed	Enabled
Is Encrypted	Disabled
Name Identifier Format	unspecified
X509 Token	
Sign Key Reference Mechanism	direct
Encryption Key Reference Mechanism	direct
Recipient Sign Key Reference Mechanism	direct
Recipient Encryption Key Reference Mechanism	direct
Is Signed	Disabled
Use PKI Path	Disabled
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Enabled
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Disabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

[Table 18-80](#) lists the configuration properties and the default settings for the `wss10_saml20_token_with_message_protection_client_template` assertion template.

Table 18-80 wss10_saml20_token_with_message_protection_client_template Configuration Properties

Name	Default Value	Type
<code>user.attributes</code>	None	Optional
<code>keystore.recipient.alias</code>	orakey	Required
<code>user.roles.include</code>	false	Optional

Table 18-80 (Cont.) wss10_saml20_token_with_message_protection_client_template Configuration Properties

Name	Default Value	Type
keystore.sig.csf.key	None	Optional
keystore.enc.csf.key	None	Optional
saml.issuer.name	www.oracle.com	Optional
csf-key	basic.credentials	Optional
subject.precedence	true	Optional
attesting.mapping.attribute	DN	Optional
saml.audience.uri	None	Optional
propagate.identity.context	None	Optional
ignore.timestamp.in.response	false	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.58 oracle/wss10_saml20_token_with_message_protection_service_template

This topic describes the `wss10_saml20_token_with_message_protection_service_template` assertion template.

Display Name: Wss10 SAML V2.0 Token with Message Protection service Assertion Template

Category: Security

Type: wss10-saml-with-certificates

Description

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. See [Table 18-79](#) for information about the settings.

Configuration

Table 18-81 lists the configuration properties and the default settings for the `wss10_saml20_token_with_message_protection_service_template` assertion template.

Table 18-81 `wss10_saml20_token_with_message_protection_service_template` Configuration Properties

Name	Default Value	Type
<code>role</code>	<code>ultimateReceiver</code>	Constant
<code>keystore.sig.csf.key</code>	None	Optional
<code>keystore.enc.csf.key</code>	None	Optional
<code>saml.trusted.issuers</code>	None	Optional
<code>propagate.identity.context</code>	None	Optional
<code>sc.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.59 oracle/

wss10_username_token_with_message_protection_client_template

This topic describes the `wss10_username_token_with_message_protection_client_template` assertion template.

Display Name: Wss10 Username Token with Message Protection client Assertion Template

Category: Security

Type: wss10-username-with-certificates

Description

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.

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 18-82 lists the settings for the `wss10_username_token_with_message_protection_client_template` assertion template.

Table 18-82 wss10_username_token_with_message_protection_client_template Settings

Name	Default Value
Username Token	
Password Type	plaintext
Creation Time Required	Disabled
Nonce Required	Disabled
Is Signed	Enabled
Is Encrypted	Enabled
X509 Token	
Sign Key Reference Mechanism	direct
Encryption Key Reference Mechanism	direct
Recipient Sign Key Reference Mechanism	direct
Recipient Encryption Key Reference Mechanism	direct
Is Signed	Disabled
Use PKI Path	Disabled
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Enabled
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Disabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

[Table 18-83](#) lists the configuration properties and the default settings for the `wss10_username_token_with_message_protection_client_template` assertion template.

Table 18-83 wss10_username_token_with_message_protection_client_template Configuration Properties

Name	Default Value	Type
csf-key	basic.credentials	Required
csf.map	None	Optional
role	ultimateReceiver	Constant
keystore.sig.csf.key	None	Optional
keystore.enc.csf.key	None	Optional
keystore.recipient.alias	orakey	Required
ignore.timestamp.in.response	false	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.60 oracle/wss10_username_token_with_message_protection_service_template

Display Name: Wss10 Username Token with Message Protection service Assertion Template

Category: Security

Type: wss10-username-with-certificates

Description

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.

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 18-82](#) for information about the settings.

Configuration

[Table 18-84](#) lists the configuration properties and the default settings for the `wss10_username_token_with_message_protection_service_template` assertion template.

Table 18-84 wss10_username_token_with_message_protection_service_template Configuration Properties

Name	Default Value	Type
role	ultimateReceiver	Constant
csf.map	None	Optional
keystore.sig.csf.key	None	Optional
keystore.enc.csf.key	None	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.61 oracle/ wss10_x509_token_with_message_protection_client_template

This topic describes the `wss10_x509_token_with_message_protection_client` template assertion template.

Display Name: Wss10 X509 Token with Message Protection client Assertion Template

Category: Security

Type: wss10-mutual-auth-with-certificates

Description

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 18-85](#) lists the settings for the `wss10_x509_token_with_message_protection_client` template assertion template.

Table 18-85 wss10_x509_token_with_message_protection_client_template Settings

Name	Default Value
X509 Token	
Sign Key Reference Mechanism	direct
Encryption Key Reference Mechanism	direct
Recipient Sign Key Reference Mechanism	direct
Recipient Encryption Key Reference Mechanism	direct
Is Signed	Disabled
Use PKI Path	Disabled
Secure Conversation	
Enabled	Disabled

Table 18-85 (Cont.) wss10_x509_token_with_message_protection_client_template Settings

Name	Default Value
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Enabled
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Disabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

[Table 18-86](#) lists the configuration properties and the default settings for the `wss10_x509_token_with_message_protection_client_template` assertion template.

Table 18-86 wss10_x509_token_with_message_protection_client_template Configuration Properties

Name	Default Value	Type
role	ultimateReceiver	Constant
csf.map	None	Optional
keystore.sig.csf.key	None	Optional
keystore.enc.csf.key	None	Optional
keystore.recipient.alias	orakey	Required
ignore.timestamp.in.response	false	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.62 oracle/wss10_x509_token_with_message_protection_service_template

This topic describes the `wss10_x509_token_with_message_protection_service_template` assertion template.

Display Name: Wss10 X509 Token with Message Protection service Assertion Template

Category: Security

Type: wss10-mutual-auth-with-certificates

Description

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 18-85](#) for information about the settings.

Configuration

[Table 18-87](#) lists the configuration properties and the default settings for the `wss10_x509_token_with_message_protection_service_template` assertion template.

Table 18-87 `wss10_x509_token_with_message_protection_service_template` Configuration Properties

Name	Default Value	Type
<code>role</code>	<code>ultimateReceiver</code>	Constant
<code>csf.map</code>	None	Optional
<code>keystore.sig.csf.key</code>	None	Optional
<code>keystore.enc.csf.key</code>	None	Optional
<code>sc.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.63 oracle/wss11_kerberos_token_over_ssl_client_template

This topic describes the `wss11_kerberos_token_over_ssl_client_template` assertion template.

Display Name: Wss11 Kerberos Token Over SSL Client Assertion Template

Category: Security

Type: wss11-kerberos-over-ssl-security

Description

The `wss11_kerberos_token_over_ssl_client_template` assertion template includes a Kerberos token in the WS-Security SOAP header in accordance with the WS-Security Kerberos Token Profile v1.1 standard. The Kerberos token is advertised as an `EndorsingSupportingToken`, and is used only for authentication and for signing the timestamp. Message protection is provided by SSL.

Settings

Table 18-88 lists the settings for the `wss11_kerberos_token_over_ssl_client_template` assertion template.

Table 18-88 `wss11_kerberos_token_over_ssl_client_template` Settings

Name	Default Value
Kerberos Token Type	
Kerberos Token Type	<code>gss-apreq-v5</code>
Transport Layer Security	
Transport Layer Security	Enabled
Transport Layer Security—Mutual Authentication Required	Disabled
Transport Layer Security—Include Timestamp	Enabled
Algorithm Suite	<code>BASIC_128</code>

Configuration

Table 18-89 lists the configuration properties and the default settings for the `wss11_kerberos_token_over_ssl_client_template` assertion template.

Table 18-89 `wss11_kerberos_token_over_ssl_client_template` Configuration Properties

Name	Default Value	Type
service.principal.name	<code>HOST/ localhost@EXAMPLE.COM</code>	Required
keytab.location	None	Optional
caller.principal.name	None	Optional
credential.delegation	<code>false</code>	Required
reference.priority	None	Optional

18.64 oracle/wss11_kerberos_token_over_ssl_service_template

This topic describes the `wss11_kerberos_token_service_template` assertion template.

Display Name: Wss11 Kerberos Token Over SSL Service Assertion Template

Category: Security

Type: `wss11-kerberos-over-ssl-security`

Description

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. The Kerberos token is advertised as an `EndorsingSupportingToken`, and is used only for authentication and for signing the timestamp. Message protection is provided by SSL.

Settings

The settings for the `wss11_kerberos_token_over_ssl_service_template` are identical to the client version of the assertion template. See [Table 18-88](#) for information about the settings.

Configuration

[Table 18-90](#) lists the configuration properties and the default settings for the `wss11_kerberos_token_over_ssl_service_template` assertion template.

Table 18-90 `wss11_kerberos_token_over_ssl_service_template` Configuration Properties

Name	Default Value	Type
<code>credential.delegation</code>	false	Required
<code>reference.priority</code>	None	Optional

18.65 oracle/wss11_kerberos_token_with_message_protection_client_template

This topic describes the `wss11_kerberos_token_with_message_protection_client_template` assertion template.

Display Name: Wss11 Kerberos Token with message protection client Assertion Template

Category: Security

Type: kerberos-security

Description

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 18-91](#) lists the settings for the `wss11_kerberos_token_with_message_protection_client_template` assertion template.

Table 18-91 wss11_kerberos_token_with_message_protection_client_template Settings

Name	Default Value
Kerberos Token Type	
Kerberos Token Type	gss-apreq-v5
Derived Keys	Disabled
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Enabled
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	TRIPLE_DES
Include Timestamp	Enabled
Confirm Signature	Enabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

[Table 18-92](#) lists the configuration properties and the default settings for the `wss11_kerberos_token_with_message_protection_client_template` assertion template.

Table 18-92 wss11_kerberos_token_with_message_protection_client_template Configuration Properties

Name	Default Value	Type
<code>service.principal.name</code>	HOST/ localhost@EXAMPLE.COM	Required
<code>keytab.location</code>	None	Optional
<code>caller.principal.name</code>	None	Optional
<code>credential.delegation</code>	false	Required
<code>sc.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.66 oracle/ wss11_kerberos_token_with_message_protection_service_template

This topic describes the `wss11_kerberos_token_with_message_protection_service_template` assertion template.

Display Name: Wss11 Kerberos Token service with message protection Assertion Template

Category: Security

Type: kerberos-security

Description

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_kerberos_token_with_message_protection_service_template` are identical to the client version of the assertion template. See [Table 18-91](#) for information about the settings.

Configuration

[Table 18-93](#) lists the configuration properties and the default settings for the `wss11_kerberos_token_with_message_protection_service_template` assertion template.

Table 18-93 `wss11_kerberos_token_with_message_protection_service_template` Configuration Properties

Name	Default Value	Type
credential.delegation	false	Required
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.67 oracle/ wss11_saml_token_with_message_protection_client_template

This topic describes the `wss11_saml_token_with_message_protection_client_template` assertion template.

Display Name: Wss11 SAML Token with Message Protection client Assertion Template

Category: Security

Type: wss11-saml-with-certificates

Description

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.

Settings

[Table 18-94](#) lists the settings for the `wss11_saml_token_with_message_protection_client_template` assertion template.

Table 18-94 `wss11_saml_token_with_message_protection_client_template` Settings

Name	Default Value
SAML Token Type	
Version	1.1
Confirmation Type	sender-vouches
Is Signed	Enabled
Is Encrypted	Disabled
Name Identifier Format	unspecified
X509 Token	
Sign Key Reference Mechanism	direct
Encryption Key Reference Mechanism	thumbprint
Is Signed	Enabled
Use PKI Path	Disabled
Derived Keys	Disabled
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Enabled
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Enabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration Properties

Table 18-95 lists the configuration properties and the default settings for the `wss11_saml_token_with_message_protection_client_template` assertion template.

Table 18-95 `wss11_saml_token_with_message_protection_client_template` Configuration Properties

Name	Default Value	Type
<code>user.attributes</code>	None	Optional
<code>saml.issuer.name</code>	<code>www.oracle.com</code>	Optional
<code>role</code>	<code>ultimateReceiver</code>	Constant
<code>keystore.recipient.alias</code>	<code>orakey</code>	Required
<code>keystore.sig.csf.key</code>	None	Optional
<code>keystore.enc.csf.key</code>	None	Optional
<code>csf.map</code>	None	Optional
<code>csf-key</code>	<code>basic.credentials</code>	Optional
<code>subject.precedence</code>	<code>true</code>	Optional
<code>saml.audience.uri</code>	None	Optional
<code>propagate.identity.context</code>	None	Optional
<code>user.tenant.name</code>	None	Optional
<code>ignore.timestamp.in.response</code>	<code>false</code>	Optional
<code>sc.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.68 oracle/

`wss11_saml_token_with_message_protection_service_template`

This topic describes the `wss11_saml_token_with_message_protection_service_template` assertion template.

Display Name: Wss11 SAML Token with Message Protection service Assertion Template

Category: Security

Type: wss11-saml-with-certificates

Description

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. See [Table 18-94](#) for information about the settings.

Configuration

[Table 18-96](#) lists the configuration properties and the default settings for the `wss11_saml_token_with_message_protection_service_template` assertion template.

Table 18-96 `wss11_saml_token_with_message_protection_service_template` Configuration Properties

Name	Default Value	Type
role	<code>ultimateReceiver</code>	Constant
keystore.enc.csf.key	None	Optional
saml.trusted.issuers	None	Optional
propagate.identity.context	None	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.69 oracle/

`wss11_saml20_token_with_message_protection_client_template`

This topic describes the `wss11_saml20_token_with_message_protection_client_template` assertion template.

Display Name: Wss11 SAML V2.0 Token with Message Protection client Assertion Template

Category: Security

Type: wss11-saml-with-certificates

Description

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 18-97](#) lists the settings for the `wss11_saml20_token_with_message_protection_client_template` assertion template.

Table 18-97 `wss11_saml20_token_with_message_protection_client_template` Settings

Name	Default Value
SAML Token Type	
Version	2.0

Table 18-97 (Cont.) wss11_saml20_token_with_message_protection_client_template Settings

Name	Default Value
Confirmation Type	sender-vouches
Is Signed	Enabled
Is Encrypted	Disabled
Name Identifier Format	unspecified
X509 Token	
Sign Key Reference Mechanism	direct
Encryption Key Reference Mechanism	thumbprint
Is Signed	Enabled
Use PKI Path	Disabled
Derived Keys	Disabled
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Enabled
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Enabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

Table 18-98 lists the configuration properties and the default settings for the wss11_saml20_token_with_message_protection_client_template assertion template.

Table 18-98 wss11_saml20_token_with_message_protection_client_template Configuration Properties

Name	Default Value	Type
user.attributes	None	Optional
saml.issuer.name	www.oracle.com	Optional
role	ultimateReceiver	Constant

Table 18-98 (Cont.) wss11_saml20_token_with_message_protection_client_template Configuration Properties

Name	Default Value	Type
keystore.recipient.alias	orakey	Required
keystore.sig.csf.key	None	Optional
keystore.enc.csf.key	None	Optional
csf-key	basic.credentials	Optional
subject.precedence	true	Optional
attesting.mapping.attribute	None	Optional
saml.audience.uri	None	Optional
propagate.identity.context	None	Optional
ignore.timestamp.in.response	false	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.70 oracle/wss11_saml20_token_with_message_protection_service_template

This topic describes the `wss11_saml20_token_with_message_protection_service_template` assertion template.

Display Name: Wss11 SAML V2.0 Token with Message Protection service Assertion Template

Category: Security

Type: wss11-saml-with-certificates

Description

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. See [Table 18-97](#) for information about the settings.

Configuration

[Table 18-99](#) lists the configuration properties and the default settings for the `wss11_saml20_token_with_message_protection_service_template` assertion template.

Table 18-99 wss11_saml20_token_with_message_protection_service_template Configuration Properties

Name	Default Value	Type
role	ultimateReceiver	Constant
keystore.enc.csf.key	None	Optional
saml.trusted.issuers	None	Optional
propagate.identity.context	None	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.71 oracle/

wss11_username_token_with_message_protection_client_template

This topic describes the `wss11_username_token_with_message_protection_client_template` assertion template.

Display Name: Wss11 Username Token with Message Protection client Assertion Template

Category: Security

Type: wss11-username-with-certificates

Description

The `wss11_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 18-100](#) lists the settings for the `wss11_username_token_with_message_protection_client_template` assertion template.

Table 18-100 wss11_username_token_with_message_protection_client_template Settings

Name	Default Value
Username Token	
Password Type	plaintext
Creation Time Required	Disabled

Table 18-100 (Cont.) wss11_username_token_with_message_protection_client_template Settings

Name	Default Value
Nonce Required	Disabled
Is Encrypted	Enabled
Is Signed	Enabled
X509 Token	
Encryption Key Reference Mechanism	thumbprint
Is Signed	Enabled
Use PKI Path	Disabled
Derived Keys	Disabled
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Enabled
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Enabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

[Table 18-101](#) lists the configuration properties and the default settings for the `wss11_username_token_with_message_protection_client_template` assertion template.

Table 18-101 wss11_username_token_with_message_protection_client_template Configuration Properties

Name	Default Value	Type
csf-key	basic.credentials	Required
role	ultimateReceiver	Constant
keystore.recipient.alias	orakey	Required
keystore.enc.csf.key	None	Optional

Table 18-101 (Cont.) wss11_username_token_with_message_protection_client_template Configuration Properties

Name	Default Value	Type
user.tenant.name	None	Optional
ignore.timestamp.in.response	false	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.72 oracle/

wss11_username_token_with_message_protection_service_template

This topic describes the `wss11_username_token_with_message_protection_service_template` assertion template.

Display Name: Wss11 Username Token with Message Protection service Assertion Template

Category: Security

Type: wss11-username-with-certificates

Description

The `wss11_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 18-100](#) for information about the settings.

Configuration

[Table 18-102](#) lists the configuration properties and the default settings for the `wss11_username_token_with_message_protection_service_template` assertion template.

Table 18-102 wss11_username_token_with_message_protection_service_template Configuration Properties

Name	Default Value	Type
role	ultimateReceiver	Constant

Table 18-102 (Cont.) wss11_username_token_with_message_protection_service_template Configuration Properties

Name	Default Value	Type
keystore.enc.csf.key	None	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.73 oracle/ wss11_x509_token_with_message_protection_client_template

This topic describes the `wss11_x509_token_with_message_protection_client_template` assertion template.

Display Name: Wss11 X509 Token with Message Protection client Assertion Template

Category: Security

Type: wss11-mutual-auth-with-certificates

Description

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 18-103](#) lists the settings for the `wss11_x509_token_with_message_protection_client_template` assertion template.

Table 18-103 wss11_x509_token_with_message_protection_client_template Settings

Name	Default Value
X509 Token	
Sign Key Reference Mechanism	direct
Encryption Key Reference Mechanism	thumbprint
Is Signed	Enabled
Use PKI Path	Disabled
Derived Keys	Disabled
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled

Table 18-103 (Cont.) wss11_x509_token_with_message_protection_client_template Settings

Name	Default Value
Derived Keys	Enabled
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Enabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

[Table 18-104](#) lists the configuration properties and the default settings for the `wss11_x509_token_with_message_protection_client_template` assertion template.

Table 18-104 wss11_x509_token_with_message_protection_client_template Configuration Properties

Name	Default Value	Type
role	<code>ultimateReceiver</code>	Constant
keystore.recipient.alias	<code>orakey</code>	Required
keystore.sig.csf.key	None	Optional
keystore.enc.csf.key	None	Optional
ignore.timestamp.in.response	<code>false</code>	Optional
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.74 oracle/

wss11_x509_token_with_message_protection_service_template

This topic describes the `wss11_x509_token_with_message_protection_service_template` assertion template.

Display Name: Wss11 X509 Token with Message Protection service Assertion Template

Category: Security

Type: wss11-mutual-auth-with-certificates

Description

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 18-103](#) for information about the settings.

Configuration

[Table 18-105](#) lists the configuration properties and the default settings for the `wss11_x509_token_with_message_protection_service_template` assertion template.

Table 18-105 `wss11_x509_token_with_message_protection_service_template` Configuration Properties

Name	Default Value	Type
<code>role</code>	<code>ultimateReceiver</code>	Constant
<code>keystore.enc.csf.key</code>	None	Optional
<code>sc.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.75 oracle/binding_oes_authorization_template

This topic describes the `binding_oes_authorization_template` assertion template.

Display Name: Binding OES Authorization Assertion Template

Category: Security

Type: oes-authorization

Description

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 endpoint.

Settings

[Table 18-106](#) lists the settings for the `binding_oes_authorization_template` assertion template.

Table 18-106 binding_oes_authorization_template Settings

Name	Default Value
OES Based Authorization	
Guard (see Permissions)	
Action Match	*
Constraint Match	None
Resource Match	*

Configuration

[Table 18-107](#) lists the configuration properties and the default settings for the `binding_oes_authorization_template` assertion template.

Table 18-107 binding_oes_authorization_template Configuration Properties

Name	Default Value	Type
<code>application.name</code>	None	Optional
<code>resource.type</code>	None	Optional
<code>resource.name</code>	None	Optional
<code>lookup.action</code>	None	Optional
<code>execute.action</code>	None	Optional
<code>use.single.step</code>	false. Does not apply to masking template.	Optional
<code>reference.priority</code>	None	Optional

18.76 oracle/binding_oes_masking_template

This topic describes the `binding_oes_masking_template` assertion template.

Display Name: Response masking using Oracle Entitlements Server.

Category: Security

Type: oes-masking

Description

The `binding_oes_masking_template` assertion template 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.

Settings

[Table 18-106](#) lists the settings for the `binding_oes_masking_template` assertion template.

Configuration

[Table 18-107](#) lists the configuration properties and the default settings for the `binding_oes_masking_template` assertion template.

18.77 oracle/component_oes_authorization_template

This topic describes the `component_oes_authorization_template` assertion template.

Display Name: Component OES Authorization Assertion Template

Category: Security

Type: oes-authorization

Description

The `component_oes_authorization_template` assertion template does user authorization based on a 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 a SCA component.

Settings

[Table 18-106](#) lists the settings for the `component_oes_authorization_template` assertion template.

Configuration

[Table 18-107](#) lists the configuration properties and the default settings for the `component_oes_authorization_template` assertion template.

18.78 oracle/pii_security_template

This topic describes the `pii_security_template` assertion template.

Display Name: PII Security Assertion Template

Category: Security

Type: pii-security

Description

The `pii_security_template` assertion template secures personally identifiable information (PII) using encryption. PII is identified by XPath configuration.



Note:

This assertion template applies to SOA and JCA adapters only.

Settings

[Table 18-108](#) lists the settings for the `pii_security_template` assertion template.

Table 18-108 pii_security_template Settings

Name	Default Value
PII Security	
algorithm	PBKDF2. This setting cannot be changed.
salt	pii-security
iteration	1000
keysize	128
encryption-algorithm	AES/CBC/PKCS5Padding. This setting cannot be changed.

Configuration

Table 18-109 lists the configuration properties and the default settings for the `pii_security_template` assertion template.

Table 18-109 pii_security_template Configuration Properties

Name	Default Value	Type
Request XPath	None	Optional
Request Namespaces	None	Optional
Response XPath	None	Optional
Response Namespaces	None	Optional
csf-key	pii-csf-key	Required
reference.priority	0	Optional

18.79 oracle/sts_trust_config_client_template

This topic describes the `oracle/sts_trust_config_client_template` assertion template.

Display Name: Trust Configuration Client Assertion Template

Category: Security

Type: sts-trust-config

Description

STS Configuration information, provided on the client side, that is used to invoke STS for token exchange.

Settings

Table 18-110 lists the settings for the `oracle/sts_trust_config_client_template` assertion template.

Table 18-110 oracle/sts_trust_config_client_template Settings

Name	Default Value
STS Configuration	

Table 18-110 (Cont.) oracle/sts_trust_config_client_template Settings

Name	Default Value
WSDL Exist	Yes
WSDL	http://host:port/sts?wsdl
Port URI	None
Service	None
Port	None
Port Endpoint	target-namespace#wSDL.endpoint(service-name/port-name)
Client Policy URI	None
Keystore Recipient Alias	sts-csf-key

Configuration

Table 18-111 lists the configuration properties and the default settings for the `oracle/sts_trust_config_client_template` assertion template.

Table 18-111 oracle/sts_trust_config_client_template Properties

Name	Default Value	Type
role	ultimateReceiver	Constant
reference.priority	None	Optional
wsdl.uri	http://host:port/sts?wsdl	Optional
port.uri	http://host:port/sts-service	Optional
port.endpoint	target-namespace#wSDL.endpoint(service-name/port-name)	Optional
policy.reference.uri	oracle/policy-name	Optional
sts.keystore.recipient.alias	sts-csf-key	Optional

18.80 oracle/sts_trust_config_service_template

This topic describes the `oracle/sts_trust_config_service_template` assertion template.

Display Name: Trust Configuration Service Assertion Template

Category: Security

Type: sts-trust-config

Description

Minimal STS Configuration information, provided on the service side, that is used to obtain all other STS information and invoke STS for token exchange.

Settings

Table 18-112 lists the settings for the `oracle/sts_trust_config_service_template` assertion template.

Table 18-112 oracle/sts_trust_config_service_template Settings

Name	Default Value
STS Configuration	
WSDL Exist	Yes
WSDL	http://host:port/sts?wsdl
Port URI	http://host:port/sts-service

Configuration

Table 18-113 lists the configuration properties and the default settings for the `oracle/sts_trust_config_service_template` assertion template.

Table 18-113 oracle/sts_trust_config_service_template Properties

Name	Default Value	Type
role	ultimateReceiver	Constant
reference.priority	None	Optional
wsdl.uri	http://host:port/sts?wsdl	Optional
port.uri	http://host:port/sts-service	Optional

18.81 oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template

This topic describes the `oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template` assertion template.

Display Name: Wss Issued Saml Bearer Token with Message Protection Client Assertion Template

Category: Security

Type: wss-sts-issued-token-over-ssl

Description

SOAP binding level policy for Issued Token SAML authentication (confirmation method as bearer) with SSL Message Protection.

Settings

Table 18-114 lists the settings for the `oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template` assertion template.

Table 18-114 oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template Settings

Name	Default Value
Issued Token	
Token Type	SAML11
Key Type	Bearer
Algorithm Suite	None
Derived Keys	Disabled
Transport Layer Security	
Transport Layer Security	Enabled
Mutual Authentication Required	Disabled
Include Timestamp	Enabled
Algorithm Suite	None
Algorithm Suite	BASIC_128
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Disabled
Server Entropy	Enabled

Configuration

Table 18-115 lists the configuration properties and the default settings for the `oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template` assertion template.

Table 18-115 oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template Properties

Name	Default Value	Type
sts.auth.user.csf.key	None	Optional
sts.auth.x509.csf.key	None	Optional
on.behalf.of	false	Required
sts.auth.on.behalf.of.csf.key	None	Optional
sts.auth.on.behalf.of.username.only	true	Optional
sts.keystore.recipient.alias	None	Optional
sts.auth.service.principal.name	HOST/ localhost@EXAMPLE.COM	Optional
sts.auth.keytab.location	None	Optional
sts.auth.caller.principal.name	None	Optional
ignore.timestamp.in.response	false	Optional

Table 18-115 (Cont.) oracle/wss_sts_issued_saml_bearer_token_over_ssl_client_template Properties

Name	Default Value	Type
sts.in.order	None	Optional
sc.token.lifetime	None	Optional
issued.token.lifetime	None	Optional
reference.priority	None	Optional

18.82 oracle/ wss_sts_issued_saml_bearer_token_over_ssl_service_template

This topic describes the `oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template` Assertion Template.

Display Name: Wss Issued Saml Bearer Token with Message Protection Service Assertion Template

Category: Security

Type: wss-sts-issued-token-over-ssl

Description

SOAP binding level policy for Issued Token SAML authentication (confirmation method as bearer) With SSL Message Protection.

Settings

The settings for the `oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template` are identical to the client version of the assertion template. See [Table 18-114](#) for information about the settings.

Configuration

[Table 18-116](#) lists the configuration properties and the default settings for the `oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template` assertion template.

Table 18-116 oracle/wss_sts_issued_saml_bearer_token_over_ssl_service_template Properties

Name	Default Value	Type
role	ultimateReceiver	Constant
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.83 oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template

This topic describes the `wss11_sts_issued_saml_hok_with_message_protection_client_template` assertion template.

Display Name: Wss11 Issued Token with Saml Holder of Key with Message Protection Client Assertion Template

Category: Security

Type: wss11-sts-issued-token-with-certificates

Description

WS-Security 1.1 Issued Token SAML HOK with Certificates. Provides Authenticates and Message Protection using Basic128.

Settings

[Table 18-117](#) lists the settings for the `wss11_sts_issued_saml_hok_with_message_protection_client_template` assertion template.

Table 18-117 oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template Settings

Name	Default Value
Issued Token	
Token Type	SAML11
Key Type	Symmetric
Algorithm Suite	Basic128
Derived Keys	Disabled
X509 Token	
Sign Key Reference Mechanism	thumbprint
Encryption Key Reference Mechanism	thumbprint
Is Signed	Enabled
Use PKI Path	Disabled
Derived Keys	Disabled
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled

Table 18-117 (Cont.) oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template Settings

Name	Default Value
Derived Keys	Enabled
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Enabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

Table 18-118 lists the configuration properties and the default settings for the `wss11_sts_issued_saml_hok_with_message_protection_client_template` assertion template.

Table 18-118 oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template Properties

Name	Default Value	Type
<code>sts.auth.user.csf.key</code>	None	Optional
<code>sts.auth.x509.csf.key</code>	<code>enc-csf-key</code>	Optional
<code>on.behalf.of</code>	<code>false</code>	Required
<code>sts.auth.on.behalf.of.csf.key</code>	None	Optional
<code>sts.auth.on.behalf.of.username.only</code>	<code>true</code>	Optional
<code>sts.keystore.recipient.alias</code>	None	Optional
<code>keystore.recipient.alias</code>	<code>orakey</code>	Required
<code>keystore.enc.csf.key</code>	None	Optional
<code>sts.auth.service.principal.name</code>	<code>HOST/ localhost@EXAMPLE.COM</code>	Optional
<code>sts.auth.keytab.location</code>	None	Optional
<code>sts.auth.caller.principal.name</code>	None	Optional
<code>ignore.timestamp.in.response</code>	<code>false</code>	Optional
<code>sts.in.order</code>	None	Optional
<code>sc.token.lifetime</code>	None	Optional
<code>issued.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.84 oracle/ wss11_sts_issued_saml_hok_with_message_protection_service _template

This topic describes the `wss11_sts_issued_saml_hok_with_message_protection_service_template` assertion template.

Display Name: Wss11 Issued Token with Saml Holder of Key with Message Protection Service Assertion Template

Category: Security

Type: wss11-sts-issued-token-with-certificates

Description

WS-Security 1.1 Issued Token SAML HOK with Certificates. Provides Authenticates and Message Protection using Basic128.

Settings

[Table 18-117](#) lists the settings for the `wss11_sts_issued_saml_hok_with_message_protection_service_template` assertion template.

Configuration

[Table 18-119](#) lists the configuration properties and the default settings for the `wss11_sts_issued_saml_hok_with_message_protection_service_template` assertion template.

Table 18-119 oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template Properties

Name	Default Value	Type
keystore.enc.csf.key	None	Optional
role	ultimateReceiver	Constant
sc.token.lifetime	None	Optional
reference.priority	None	Optional

18.85 oracle/ wss11_sts_issued_saml_with_message_protection_client_template

This topic describes the `wss11_sts_issued_saml_with_message_protection_client_template` assertion template.

Display Name: Wss11 Issued Token Saml Sender Voucher with Message Protection Client Assertion Template

Category: Security

Type: wss11-sts-issued-token-with-certificates

Description

WS-Security 1.1 Issued Token SAML Sender Voucher with Certificates. Provides Authenticates and Message Protection using Basic128.

Settings

[Table 18-120](#) lists the settings for the `wss11_sts_issued_saml_with_message_protection_client_template` assertion template.

Table 18-120 `wss11_sts_issued_saml_with_message_protection_client_template` Settings

Name	Default Value
Issued Token	
Token Type	SAML11
Key Type	None
Algorithm Suite	Basic128
Derived Keys	Disabled
X509 Token	
Sign Key Reference Mechanism	direct
Encryption Key Reference Mechanism	thumbprint
Is Signed	Enabled
Use PKI Path	Disabled
Derived Keys	Disabled
Secure Conversation	
Enabled	Disabled
Version	1.3 or 1.4. OWSM WS-SC supports both Secure Conversation version 1.3 and 1.4. Although the policy displays the 1.3 version number, you use this policy for 1.4 as well.
Re-authenticate	Disabled
Client Entropy	Enabled
Derived Keys	Enabled

Table 18-120 (Cont.) wss11_sts_issued_saml_with_message_protection_client_template Settings

Name	Default Value
Server Entropy	Enabled
Bootstrap Message Security	Inherit from Application Setting
Message Security	
Algorithm Suite	BASIC_128
Include Timestamp	Enabled
Confirm Signature	Enabled
Encrypt Signature	Disabled
Request Message Settings	See Table 18-131
Response Message Settings	See Table 18-131
Fault Message Settings	See Table 18-131

Configuration

Table 18-121 lists the configuration properties and the default settings for the `wss11_sts_issued_saml_with_message_protection_client_template` assertion template.

Table 18-121 oracle/wss11_sts_issued_saml_with_message_protection_client_template Properties

Name	Default Value	Type
<code>sts.auth.user.csf.key</code>	None	Optional
<code>sts.auth.x509.csf.key</code>	None	Optional
<code>on.behalf.of</code>	true	Required
<code>sts.auth.on.behalf.of.csf.key</code>	None	Optional
<code>sts.auth.on.behalf.of.username.only</code>	true	Optional
<code>sts.keystore.recipient.alias</code>	None	Optional
<code>keystore.recipient.alias</code>	orakey	Optional
<code>keystore.enc.csf.key</code>	None	Optional
<code>sts.in.order</code>	None	Optional
<code>sc.token.lifetime</code>	None	Optional
<code>ignore.timestamp.in.response</code>	false	Optional
<code>issued.token.lifetime</code>	None	Optional
<code>reference.priority</code>	None	Optional

18.86 oracle/binding_authorization_template

This topic describes the `binding_authorization_template` assertion template.

Display Name: Binding Authorization Assertion Template

Category: Security

Type: binding-authorization

Description

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 18-122 lists the settings for the `binding_authorization_template` assertion template.

Table 18-122 binding_authorization_template Settings

Name	Default Value
Authorization Permission	
Permissions— Action Match	None
Permissions— Constraint Match	None
Authorization Permission	
Guard (see Permissions)	
Action Match	None
Constraint Match	None
Resource Match	None
Roles	Not Set

Configuration

Table 18-123 lists the configuration properties and the default settings for the `binding_authorization_template` assertion template.

Table 18-123 binding_authorization_template Properties

Name	Default Value	Type
reference.priority	None	Optional

18.87 oracle/binding_permission_authorization_template

This topic describes the `binding_permission_authorization_template` assertion template.

Display Name: Binding Permission Based Authorization Assertion Template

Category: Security

Type: binding-permission-authorization

Description

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 18-124 lists the settings for the `binding_permission_authorization_template` assertion template.

Table 18-124 `binding_permission_authorization_template` Settings

Name	Default Value
Authorization Permission	
Guard (see Permissions)	
Action Match	*
Constraint Match	None
Resource Match	*
Check Permission	
Permission Class	None

Configuration

Table 18-125 lists the configuration properties and the default settings for the `binding_permission_authorization_template` assertion template.

Table 18-125 `binding_permission_authorization_template` Properties

Name	Default Value	Type
reference.priority	None	Optional

18.88 oracle/component_authorization_template

This topic describes the `component_authorization_template` assertion template.

Display Name: Component Authorization Assertion Template

Category: Security

Type: sca-component-authorization

Description

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 18-126 lists the settings for the `component_authorization_template` assertion template.

Table 18-126 `component_authorization_template` Settings

Name	Default Value
Authorization Permission	

Table 18-126 (Cont.) component_authorization_template Settings

Name	Default Value
Guard (see Permissions)	
Action Match	None
Constraint Match	None
Resource Match	None
Roles	Not Set

Configuration

[Table 18-127](#) lists the configuration properties and the default settings for the `component_authorization_template` assertion template.

Table 18-127 component_authorization_template Properties

Name	Default Value	Type
reference.priority	None	Optional

18.89 oracle/component_permission_authorization_template

This topic describes the `component_permission_authorization_template` assertion template.

Display Name: Component Permission Based Authorization Assertion Template

Category: Security

Type: sca-component-permission-authorization

Description

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 18-128](#) lists the settings for the `component_permission_authorization_template` assertion template.

Table 18-128 component_permission_authorization_template Settings

Name	Default Value
Authorization Permission	
Guard (see Permissions)	
Action Match	*
Constraint Match	None
Resource Match	None
Permission Class	None

Configuration

[Table 18-129](#) lists the configuration properties and the default settings for the component_permission_authorization_template assertion template.

Table 18-129 component_permission_authorization_template Properties

Name	Default Value	Type
reference.priority	None	Optional

18.90 Supported Algorithm Suites

[Table 18-130](#) 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 with:

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/>



Note:

FIPS compliant algorithm suites are marked with an asterisk (*). See "Enabling FIPS Mode" in *Administering Security for Oracle WebLogic Server 14c (14.1.2)* for FIPS information.

Table 18-130 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
Basic256Sha256	Sha256	Aes256	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256	HmacSha1	RsaSha1
Basic192Sha256	Sha256	Aes192	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192	HmacSha1	RsaSha1
Basic128Sha256	Sha256	Aes128	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128	HmacSha1	RsaSha1
TripleDesSha256	Sha256	TripleDes	KwTripleDes	KwRsaOaep	PSha1L192	PSha1L192	192	HmacSha1	RsaSha1
Basic256Sha256Rsa15	Sha256	Aes256	KwAes256	KwRsa15	PSha1L256	PSha1L192	256	HmacSha1	RsaSha1
Basic192Sha256Rsa15	Sha256	Aes192	KwAes192	KwRsa15	PSha1L192	PSha1L192	192	HmacSha1	RsaSha1
Basic128Sha256Rsa15	Sha256	Aes128	KwAes128	KwRsa15	PSha1L128	PSha1L128	128	HmacSha1	RsaSha1
TripleDesSha256Rsa15	Sha256	TripleDes	KwTripleDes	KwRsa15	PSha1L192	PSha1L192	192	HmacSha1	RsaSha1
Basic256Exn256	Sha256	Aes256	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256	HmacSha256	RsaSha256

Table 18-130 (Cont.) 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
Basic192Exn256	Sha256	Aes192	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192	HmacSha256	RsaSha256
Basic128Exn256	Sha256	Aes128	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128	HmacSha256	RsaSha256
TripleDesExn256	Sha256	TripleDes	KwTripleDes	KwRsaOaep	PSha1L192	PSha1L192	192	HmacSha256	RsaSha256
Basic256Exn256Rsa15*	Sha256	Aes256	KwAes256	KwRsa15	PSha1L256	PSha1L192	256	HmacSha256	RsaSha256
Basic192Exn256Rsa15*	Sha256	Aes192	KwAes192	KwRsa15	PSha1L192	PSha1L192	192	HmacSha256	RsaSha256
Basic128Exn256Rsa15*	Sha256	Aes128	KwAes128	KwRsa15	PSha1L128	PSha1L128	128	HmacSha256	RsaSha256
TripleDesExn256Rsa15*	Sha256	TripleDes	KwTripleDes	KwRsa15	PSha1L192	PSha1L192	192	HmacSha256	RsaSha256
Basic256GCM	Sha1	Aes256GCM	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256	HmacSha1	RsaSha1
Basic192GCM	Sha1	Aes192GCM	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192	HmacSha1	RsaSha1
Basic128GCM	Sha1	Aes128GCM	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128	HmacSha1	RsaSha1
Basic256GCMRsa15	Sha1	Aes256GCM	KwAes256	KwRsa15	PSha1L256	PSha1L192	256	HmacSha1	RsaSha1
Basic192GCMRsa15	Sha1	Aes192GCM	KwAes192	KwRsa15	PSha1L192	PSha1L192	192	HmacSha1	RsaSha1
Basic128GCMRsa15	Sha1	Aes128GCM	KwAes128	KwRsa15	PSha1L128	PSha1L128	128	HmacSha1	RsaSha1
Basic256GCMExn256	Sha256	Aes256GCM	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256	HmacSha256	RsaSha256
Basic192GCMExn256	Sha256	Aes192GCM	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192	HmacSha256	RsaSha256
Basic128GCMExn256	Sha256	Aes128GCM	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128	HmacSha256	RsaSha256
Basic256GCMExn256Rsa15	Sha256	Aes256GCM	KwAes256	KwRsa15	PSha1L256	PSha1L192	256	HmacSha256	RsaSha256
Basic192GCMExn256Rsa15	Sha256	Aes192GCM	KwAes192	KwRsa15	PSha1L192	PSha1L192	192	HmacSha256	RsaSha256
Basic128GCMExn256Rsa15	Sha256	Aes128GCM	KwAes128	KwRsa15	PSha1L128	PSha1L128	128	HmacSha256	RsaSha256

 **Note:**

To use the extended algorithm suites for Symmetric Signature HmacSha256 and Asymmetric Signature RsaSha256, you need to create a custom OWSM policy with algorithm suite set to extended algorithm suite type. For instance, if the algorithm suite type is `Basic256Exn256`, then it should be set as follows:

```
orasp:algorithm-suite="Basic256Exn256"
```

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*.

18.91 Message Signing and Encryption Settings for Request, Response, and Fault Messages

Table 18-131 lists the settings for the Request, Response, and Fault messages. You configure these settings for message signing and encryption.

Table 18-131 Request, Response, and Fault Message Signing and Encryption Settings

Name	Default Value
Include Entire Body	True for Request and Response messages False for Fault messages
Include SwA Attachment	False
Include MIME Headers	False
Header Elements	None
Body Elements	None

18.92 oracle/security_log_template

You can use the `security_log_template` assertion template for debugging and auditing purposes.

Display Name: Security Log Assertion Template

Category: Security

Type: Logging

Description

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 18-132](#) lists the settings for the `security_log_template` assertion template.

Table 18-132 security_log_template Settings

Name	Default Value
Logging	
Request	all
Response	soap_body
Fault	Not set

Configuration

[Table 18-133](#) lists the configuration properties and the default settings for the `security_log_template` assertion template.

Table 18-133 security_log_template Properties

Name	Default Value	Type
reference.priority	None	Optional

Part VI

Security and Policy Reference for Oracle Web Services

Learn about web service security and policy annotations; assertion and policy set schemas; and the OWSM plug-in for Oracle Virtual Assembly Builder.

Part VI provides reference information that describes web service security and policy annotations; assertion and policy set schemas; and the OWSM plug-in for Oracle Virtual Assembly Builder.

It contains the following appendices:

- [Security and Policy Annotations for Oracle Web Services](#), describes the asynchronous web service and policy annotations that are used by Oracle Infrastructure web services.
- [Predefined Assertion Templates for Oracle Web Services](#), provides details on all the assertion template settings and configuration properties.
- [Schema Reference for Predefined Assertions for Oracle Web Services](#), provides the XML schema for reference when creating a WS-Policy file that contains web service assertions.
- [Schema Reference for Web Services Policy Sets](#), provides the XML schema for reference when creating a policy set file.
- [Oracle Web Services Manager Introspection Plug-in for Oracle Virtual Assembly Builder](#), provides information about using the OWSM introspection plug-in for Oracle Virtual Assembly Builder.

Part VII

Web Server Manager Support for Web Logic Server Secure Mode

This part provides information on the SSL configuration requirements for OWSM.

- [SSL Configuration Requirements for OWSM](#) , provides configuration steps to be set in the SSL environment.

19

SSL Configuration Requirements for OWSM

Learn about SSL configurations required for Oracle Web Services Manager (OWSM). It includes the following topics:

- [WSM support for WLS Secure Mode](#)

19.1 WSM support for WLS Secure Mode

On the Fusion Middleware (FMW) domain, following configurations must be set in the SSL environment.

1. Check CCW configuration on the domain.
 - a. Log into Enterprise Manager (EM).
 - b. Check if wsm-pm URL is a t3s URL. To check this
 - i. Navigate to: **Weblogic Domain > Cross Component Wiring > Service Tables**
 - ii. Check for OWSM Policy Manager (wsm-pm) URL. It must be t3s.

 **Note:**

When SSL is enabled, the wsm-pm URL must not be a t3s URL.

- c. Follow step is a workaround to add t3s and https URLs.
 - i. Update `<domain_home>/config/fmwconfig/wsm-config.xml`.
 - ii. Update the `pm.url` attribute to add t3s and https URLs.

```
<orares:property orares:category="ConfigManager"
orares:name="pm.url">
  <orares:value>t3s://<hostname>:<port></orares:value>
  <orares:value>https://<hostname>:<port></orares:value>
```

2. Set the bootstrap properties for WSM. The following `wlst` should be run during server configuration or after server start-up

```
setWSMBootstrapConfig('<domain_name>', '<domain_home_dir>', 'ConfigManager', '
pm.url', 'auto-ssl')
```

3. Log into EM . Update WSM Configuration to set SSL mode on.
 - a. Navigate to **Weblogic Domain > Web Services > WSM Domain Configuration**.
 - b. Set SSL mode on, on the WSM Configuration. To do so:
 - i. Navigate to **Policy Accessor** tab.

- ii. Add PM CSF Key to point to existing key in keystore.

```
setWSMConfiguration('/WLS/  
base_domain','ConfigManager','pm.csf.key',None,['fad'])
```

- iii. Select Use SSL only option.

```
setWSMConfiguration('/WLS/  
base_domain','ConfigManager','pm.url',None,['auto-ssl'])
```

- iv. Under SSL Setup

- Select Oneway OR Two-way

```
setWSMConfiguration('/WLS/  
base_domain','ConfigManager','ssl.twoway',None,['true'])
```

- Select the Keystore (for example, KSS)

```
setWSMConfiguration('/WLS/  
base_domain','ConfigManager','truststore.csf.key',None,['fad'])
```

```
setWSMConfiguration('/WLS/  
base_domain','ConfigManager','keystore.ssl.alias',None,['fad'])
```

- Select the Truststore Path (location of keystore)

```
setWSMConfiguration('/WLS/  
base_domain','ConfigManager','truststore.path',None,['kss://owsm/  
keystore'])
```

- v. Click **Apply**.
- vi. Click **Refresh** button.

4. For the configurations changes to reflect:

- Automatic Refresh Wait for 10 min
Or
- Restart the servers to see the immediate effect of the configuration change.

For more information on secure mode, see Using Secured Production Mode.

A

Security and Policy Annotations for Oracle Web Services

Web service security and policy annotations are available to secure and enable advanced features on Oracle web services.

This appendix describes the web service security and policy annotations and includes the following sections:

- [About Security and Policy Annotations for Web Services](#)
- [Summary of Security and Policy Annotations for Web Services](#)
- [List of Security and Policy Annotations for Web Services](#)

A.1 About Security and Policy Annotations for Web Services

The Oracle web services programming model uses JDK metadata annotations.

JDK metadata annotations are described in the following technical note:

<http://docs.oracle.com/javase/7/docs/technotes/guides/language/annotations.html>

For additional information, see JSR 175 at:

<http://www.jcp.org/en/jsr/detail?id=175>

In this programming model, you create an annotated Java file to specify the shape and characteristics of the web service.

For more information about the annotations available, see *Java API Reference for Oracle Infrastructure Web Services*. For more information about the OWSM predefined policies, see [Oracle Web Services Manager Predefined Policies](#).

A.2 Summary of Security and Policy Annotations for Web Services

Web service security and policy annotations secure and enable advanced features on Oracle web services.

[Table A-1](#) summarizes the web service security and policy annotations described in this appendix. This table also lists the corresponding web service Feature class that can be used to attach policies to a subset of web service clients, as described in "[Understanding Attaching Policies to Web Services and Clients at Design Time](#)".

Table A-1 Security and Policy Annotations for Oracle Web Services

Annotation	Description
@Addressing	Specifies the use of WS-Addressing with either the SOAP 1.1/HTTP or SOAP 1.2/HTTP binding.

Table A-1 (Cont.) Security and Policy Annotations for Oracle Web Services

Annotation	Description
@AtomicTransaction	Enables web services atomic transactions.
@Buffering	Enables buffering of a Web service.
@CacheBinaryContent	Enables and configures binary caching of content.
@CallbackManagementPolicy	Attaches a management policy when sending the asynchronous response to the client callback service.
@CallbackMtomPolicy	Attaches an MTOM policy when sending the asynchronous response to the client callback service.
@CallbackPolicySet	Attaches one or more policy sets to the callback client of the asynchronous web service that will connect to the callback service.
@CallbackSecurityPolicy	Attaches a security policy when sending the asynchronous response to the client callback service.
@FastInfosetCallbackClient	Enables and configures Fast Infoset on callback client of the asynchronous web service that will connect to the callback service.
@FastInfosetClient	Enables and configures Fast Infoset on a Web service client.
@FastInfosetService	Enables Fast Infoset on a web service.
@JMSTransportClient	Enables and configures SOAP over JMS transport for JAX-WS web service clients.
@JMSTransportService	Enables and configures SOAP over JMS transport for JAX-WS web services.
@ManagementPolicy	Attaches a management policy to the web service.
@MaxRequestSize	Configures the maximum size, in bytes, of the request message that can be sent to the web service.
@MEXRequestProcessingService	Enables the exchange of metadata.
@MTOM	Enables the use of Message Transmission Optimization Mechanism (MTOM) on the web service.
@MTOMEncodeFaultService	Enables the creation of MTOM-enabled SOAP fault messages when MTOM is enabled.
@MtomPolicy	Attaches an MTOM policy to the web service.
@ OAuth1 Client Policy	Attaches a management policy which allows applications to use Twitter API using the statically generated consumer and access tokens.
@Persistence	Configures the secure conversation session persistence mechanism for the web service.
@PolicyReference	Attaches a single policy to a subject, and optionally overrides configuration property values.
@PolicySet	Defines a set of policy references, and optionally overrides unscoped configuration property values.
@POXHttpBindingService	Enables an endpoint to receive non-SOAP XML messages that are processed by a user defined <code>javax.xml.ws.Provider<T>.invoke</code> method.
@Property	Specifies a single property that can be used to override the configuration of one or more policies.
@ReliabilityPolicy	Attaches the <code>oracle/wsrml0_policy</code> or <code>oracle/wsrml0_policy reliable</code> messaging policies to the web service.
@ReliableMessaging	Attaches the <code>oracle/reliable_messaging_policy</code> policy to the web service

Table A-1 (Cont.) Security and Policy Annotations for Oracle Web Services

Annotation	Description
@RequestProcessingService	Enables the web service endpoint.
@SchemaValidation	Enables validation of request messages against the schema.
@SecurityPolicies (Oracle Infrastructure Web Services)	Specifies an array of <code>oracle.webservices.annotations.SecurityPolicy</code> annotations.
@SecurityPolicies (Java EE Web Services)	Specifies an array of <code>weblogic.wsee.jws.jaxws.owsm.SecurityPolicy</code> annotations.
@SecurityPolicy (Oracle Infrastructure Web Services)	Attaches a security policy to the request or response SOAP message.
@SecurityPolicy (Java EE Web Services)	Attaches a security policy to the request or response SOAP message.
@SOAPRequestProcessingService	Enables the processing of SOAP requests on a web service endpoint.
@TestPageProcessingService	Enables the Web Service Test Client.
@WSDLRequestProcessingService	Enables the WSDL for the web service.
@WSLoggingLevel	Sets the logging level for diagnostic logs for the web service endpoint.

A.3 List of Security and Policy Annotations for Web Services

Oracle web services includes the following security and policy annotations:

- [@Addressing](#)
- [@AtomicTransaction](#)
- [@CacheBinaryContent](#)
- [@CallbackManagementPolicy](#)
- [@CallbackMtomPolicy](#)
- [@CallbackPolicySet](#)
- [@CallbackSecurityPolicy](#)
- [@FastInfosetCallbackClient](#)
- [@FastInfosetService](#)
- [@JMSTransportClient](#)
- [@JMSTransportService](#)
- [@ManagementPolicy](#)
- [@MaxRequestSize](#)
- [@MEXRequestProcessingService](#)
- [@MTOM](#)
- [@MTOMEncodeFaultService](#)
- [@MtomPolicy](#)
- [@OAuth1 Client Policy](#)
- [@Persistence](#)
- [@PolicyReference](#)

- [@PolicySet](#)
- [@POXHttpBindingService](#)
- [@Property](#)
- [@ReliabilityPolicy](#)
- [@ReliableMessaging](#)
- [@RequestProcessingService](#)
- [@SchemaValidation](#)
- [@SecurityPolicies \(Oracle Infrastructure Web Services\)](#)
- [@SecurityPolicies \(Java EE Web Services\)](#)
- [@SOAPRequestProcessingService](#)
- [@TestPageProcessingService](#)
- [@WSDLRequestProcessingService](#)
- [@WSLoggingLevel](#)

A.3.1 @Addressing

The `javax.xml.ws.soap.Addressing` annotation specifies the use of WS-Addressing with either the SOAP 1.1/HTTP or SOAP 1.2/HTTP binding.

A.3.1.1 @Addressing Attributes

The following table defines the attributes that can be passed to the `javax.xml.ws.soap.Addressing` annotation.

Table A-2 Attributes for `javax.xml.ws.soap.Addressing` Annotation

Attribute	Description	Default
<code>enabled</code>	Boolean value that specifies whether the endpoint supports WS-Addressing.	<code>true</code>
<code>required</code>	Boolean value that specifies whether WS-Addressing headers are required to be present on the incoming message.	<code>false</code>
<code>responses</code>	Value that specifies whether the endpoint requires the use of anonymous, non-anonymous, or all types of responses. Valid values, defined by <code>javax.xml.ws.soap.AddressingFeature.Responses</code> , include: <ul style="list-style-type: none"> • <code>ALL</code> • <code>ANONYMOUS</code> • <code>NON_ANONYMOUS</code> 	<code>ALL</code>

A.3.1.2 @Addressing Example

```
@Addressing(
    enabled = true,
    required = true,
    responses = AddressingFeature.Responses.ALL
)
```


A.3.2 @AtomicTransaction

The `com.oracle.webservices.api.tx.at.AtomicTransaction` annotation enables web services atomic transactions.

For more information, see "Using Web Services Atomic Transactions" in *Developing Oracle Infrastructure Web Services*.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.2.1 @AtomicTransaction Attributes

The following table defines the attributes that can be passed to the `com.oracle.webservices.api.tx.at.AtomicTransaction` annotation.

Table A-3 Attributes for `com.oracle.webservices.api.tx.at.AtomicTransaction` Annotation

Attribute	Description	Default
enabled	Boolean value that specifies whether or not the feature is enabled.	true
flowType	Whether the web services atomic transaction coordination context is passed with the transaction flow. Valid values, defined by <code>com.oracle.webservices.api.tx.at.AtomicTransactionFlowType</code> , include: <ul style="list-style-type: none"> MANDATORY NEVER SUPPORTS For more information about the valid values, see "Configuring Web Service Atomic Transactions" in <i>Developing Oracle Infrastructure Web Services</i> .	SUPPORTS
version	Version of the web services atomic transaction coordination context that is supported for the SOA service or reference. For SOA references, it specifies the version used for outbound messages only. The value specified must be consistent across the entire transaction. Valid values, defined by <code>com.oracle.webservices.api.tx.at.AtomicTransactionVersion</code> , include: <ul style="list-style-type: none"> DEFAULT WSAT10 WSAT11 WSAT12 For more information about the valid values, see "Configuring Web Service Atomic Transactions" in <i>Developing Oracle Infrastructure Web Services</i> .	DEFAULT

A.3.2.2 @AtomicTransaction Example

```
@AtomicTransaction(
    enabled=true,
    flowType = AtomicTransactionFlowType.MANDATORY,
    version= AtomicTransactionVersion.DEFAULT)
```

A.3.3 @Buffering

The `com.oracle.webservices.api.Buffering` annotation enables buffering of a Oracle Infrastructure Web service.



Note:

This annotation applies to Oracle Infrastructure Web services only.

When an operation on a buffered Web service is invoked, the message representing that invocation is stored in a JMS queue. WebLogic Server processes this buffered message asynchronously. If WebLogic Server goes down while the message is still in the queue, it will be processed as soon as WebLogic Server is restarted.

For example:

```
@Buffering(
    enabled=true,
    requestQueueEnabled=true)
```

The following table defines the attributes that can be passed to the `com.oracle.webservices.api.Buffering` annotation.

Table A-4 Attributes for `com.oracle.webservices.api.Buffering` Annotation

Attribute	Description	Default
<code>enabled</code>	Boolean value that specifies whether or not the policy is enabled.	<code>true</code>
<code>requestQueueConnectionFactoryJNDIName</code>	JNDI name of the connection factory to use for request message buffering. This value defaults to the default JMS connection factory defined by the server.	Default JMS connection factory defined by the server.
<code>requestQueueEnabled</code>	Flag that specifies whether the request queue is enabled.	<code>false</code>
<code>responseQueueName</code>	JNDI name of the request buffering queue.	""
<code>requestQueueTransactionEnabled</code>	Flag that specifies whether transactions should be used when storing and retrieving messages from the request buffering queue.	<code>false</code>
<code>responseQueueConnectionFactoryJNDIName</code>	JNDI name of the connection factory to use for response message buffering.	Default JMS connection factory defined by the server.
<code>responseQueueEnabled</code>	Flag that specifies whether the response queue is enabled.	<code>false</code>

Table A-4 (Cont.) Attributes for com.oracle.webservices.api.Buffering Annotation

Attribute	Description	Default
responseQueueName	JNDI name of the response buffering queue.	""
responseQueueTransactionEnabled	Flag that specifies whether transactions should be used when storing and retrieving messages from the response buffering queue.	false
retryCount	Number of times that the JMS queue attempts to deliver the message to the Web service implementation until the operation is successfully invoked.	3L
retryDelay	Amount of time between retries of a buffered request and response. Note, this value is only applicable when <code>retryCount</code> is greater than 0. The value specified must be a positive value and conform to the XML schema duration lexical format, <code>PnYnMnDtnHnMnS</code> , where <code>nY</code> specifies the number of years, <code>nM</code> specifies the number of months, <code>nD</code> specifies the number of days, <code>T</code> is the date/time separator, <code>nH</code> specifies the number of hours, <code>nM</code> specifies the number of minutes, and <code>nS</code> specifies the number of seconds.	PODT30S (30 seconds)

A.3.4 @CacheBinaryContent

The `com.oracle.webservices.api.CacheBinaryContent` annotation enables and configures binary caching of content.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.4.1 @CacheBinaryContent Attributes

The following table defines the attributes that can be passed to the `com.oracle.webservices.api.CacheBinaryContent` annotation.

Table A-5 Attributes for com.oracle.webservices.api.CacheBinaryContent Annotation

Attribute	Description	Default
arg1	Boolean value that defines one of the following values: <ul style="list-style-type: none"> If <code>mode</code> is set to <code>BINARY</code>, this argument is not required. If <code>mode</code> is set to <code>FILE</code>, specifies the directory in which to store the temporary files as <code>arg1</code>. If <code>mode</code> is set to <code>BLOB</code>, specifies the URL of the DBMS connection. 	"java.io.tmpdir"
enabled	Boolean value that specifies whether or not the feature is enabled.	true

Table A-5 (Cont.) Attributes for com.oracle.webservices.api.CacheBinaryContent Annotation

Attribute	Description	Default
mode	<p>Value that specifies the runtime requirements of XTI scalable DOM in OraSAAJ. Valid values, defined by <code>com.oracle.webservices.api.CacheBinaryContentMode</code>, include:</p> <ul style="list-style-type: none"> <code>BINARY</code>—Fastest method, but most memory intensive. Not recommended for production. <code>FILE</code>—Recommended method. Specifies one temporary file per document. Need to specify the directory in which to store the temporary files as <code>arg1</code>. <code>BLOB</code>—Slowest method. Need to specify the URL of the DBMS connection as <code>arg1</code>. 	BINARY

A.3.4.2 @CacheBinaryContent Example

```
@CacheBinaryContent(
    enabled=true,
    mode= CacheBinaryContentMode.FILE,
    arg1="/mytempdir")
```

A.3.5 @CallbackManagementPolicy

The `oracle.webservices.annotations.CallbackManagementPolicy` annotation enables you to attach a management policy when sending the asynchronous response to the client callback service.

This annotation is applicable to asynchronous web service implementation classes that are annotated with the `oracle.webservices.annotations.async.AsyncWebService` annotation, as described in "Developing an Asynchronous Web Service" in *Developing Oracle Infrastructure Web Services*.



Note:

This annotation has been deprecated. Oracle recommends that you use the `oracle.wsm.metadata.annotation.CallbackPolicySet` annotation, as described in "[@CallbackPolicySet](#)".

This annotation applies to Oracle Infrastructure web services only.

A.3.5.1 @CallbackManagementPolicy Attributes

The following table defines the attributes that can be passed to the `oracle.webservices.annotations.async.CallbackManagementPolicy` annotation.

Table A-6 Attributes for oracle.webservices.annotations.async.CallbackManagementPolicy Annotation

Attribute	Description	Default
enabled	Boolean value that specifies whether or not the policy is enabled.	true

Table A-6 (Cont.) Attributes for oracle.webservices.annotations.async.CallbackManagementPolicy Annotation

Attribute	Description	Default
value	Location from which to retrieve the WS-Policy file. Use the <code>http:</code> prefix to specify the URL of a WS-Policy file on the Web. Use the <code>policy:</code> prefix to specify that the WS-Policy file is packaged in the policy repository.	""

A.3.5.2 @CallbackManagementPolicy Example

```
@CallbackManagementPolicy(
    value="oracle/log_policy",
    enabled = true
)
```

A.3.6 @CallbackMtomPolicy

The `oracle.webservices.annotations.async.CallbackMtomPolicy` annotation attaches an MTOM policy when sending the asynchronous response to the client callback service.



Note:

This annotation has been deprecated. Oracle recommends that you use the `oracle.wsm.metadata.annotation.CallbackPolicySet` annotation, as described in "[@CallbackPolicySet](#)".

This annotation applies to Oracle Infrastructure web services only.

This annotation is applicable to asynchronous web service implementation classes that are annotated with the `oracle.webservices.annotations.async.AsyncWebService` annotation, as described in "Developing an Asynchronous Web Service" in *Developing Oracle Infrastructure Web Services*.

A.3.6.1 @CallbackMtomPolicy Attributes

The following table defines the attributes that can be passed to the `oracle.webservices.annotations.async.CallbackMtomPolicy` annotation.

Table A-7 Attributes for oracle.webservices.annotations.async.CallbackMtomPolicy Annotation

Attribute	Description	Default
enabled	Boolean value that specifies whether or not the policy is enabled.	true
value	Location from which to retrieve the MTOM policy file. Use the <code>http:</code> prefix to specify the URL of a MTOM policy file on the Web. Use the <code>policy:</code> prefix to specify that the MTOM policy file is packaged in the policy repository.	""

A.3.6.2 @CallbackMtomPolicy Example

```
@CallbackMtomPolicy(
    value="oracle/wsmtom_policy",
    enabled = true
)
```

A.3.7 @CallbackPolicySet

The `oracle.wsm.metadata.annotation.CallbackPolicySet` annotation defines a set of policy references for the callback service, and optionally overrides unscoped configuration property values.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.7.1 @CallbackPolicySet Attributes

The following table defines the attributes that can be passed to the `oracle.wsm.metadata.annotation.CallbackPolicySet` annotation.

Table A-8 Attributes for `oracle.wsm.metadata.annotation.CallbackPolicySet` Annotation

Attribute	Description	Default
<code>properties</code>	List of <code>oracle.wsm.metadata.annotation.Property</code> annotations that define configuration override property values.	""
<code>references</code>	List of <code>oracle.wsm.metadata.annotation.PolicyReference</code> annotations that define policies to attach to the subject.	""

A.3.7.2 @CallbackPolicySet Example

```
@CallbackPolicySet(references = {
    @PolicyReference("oracle/wss_http_token_service_policy")
})
```

A.3.8 @CallbackSecurityPolicy

The `oracle.webservices.annotations.async.CallbackSecurityPolicy` annotation attaches a security policy when sending the asynchronous response to the client callback service.



Note:

This annotation has been deprecated. Oracle recommends that you use the `oracle.wsm.metadata.annotation.CallbackPolicySet` annotation, as described in "[@CallbackPolicySet](#)".

This annotation applies to Oracle Infrastructure web services only.

This annotation is applicable to asynchronous web service implementation classes that are annotated with the `oracle.webservices.annotations.async.AsyncWebService` annotation, as described in "Developing an Asynchronous Web Service" in *Developing Oracle Infrastructure Web Services*.

A.3.8.1 @CallbackSecurityPolicy Attributes

The following table summarizes the attributes that you can pass to the `oracle.webservices.annotations.async.CallbackSecurityPolicy` annotation.

Table A-9 Attributes for `oracle.webservices.annotations.async.CallbackSecurityPolicy` Annotation

Attribute	Description	Default
<code>enabled</code>	Optional. Boolean value that specifies whether the policy is enabled.	<code>true</code>
<code>Properties</code>	Optional. Array of property value-name pairs.	<code>""</code>
<code>value</code>	Location from which to retrieve the WS-Policy file. Use the <code>http:</code> prefix to specify the URL of a WS-Policy file on the Web. Use the <code>policy:</code> prefix to specify that the WS-Policy file is packaged in the policy repository.	<code>""</code>

A.3.8.2 @CallbackSecurityPolicy Example

```
@CallbackSecurityPolicy(value=
    "policy:oracle/wss10_username_token_with_message_protection_server_policy"),
```

A.3.9 @FastInfosetContentCallbackClient

The `com.oracle.webservices.api.FastInfosetContentCallbackClient` annotation enables and configures Fast InfosetContent on the callback client of the asynchronous web service that will connect to the callback service.



Note:

This annotation applies to Oracle Infrastructure web services only.

For more information about developing asynchronous web services and callback clients, see "Developing Asynchronous Web Services" in *Developing Oracle Infrastructure Web Services*. For more information about Fast InfosetContent, see "Optimizing XML Transmission Using Fast InfosetContent" in *Developing Oracle Infrastructure Web Services*.

A.3.9.1 @FastInfosetContentCallbackClient Attributes

The following table defines the attributes that can be passed to the `com.oracle.webservices.api.FastInfosetContentCallbackClient` annotation.

Table A-10 Attributes for `com.oracle.webservices.api.FastInfosetContentCallbackClient` Annotation

Attribute	Description	Default
<code>fastInfosetContentNegotiation</code>	<p>Content negotiation strategy. Valid values, defined by <code>com.oracle.webservices.api.FastInfosetContentNegotiationStrategy</code>, include:</p> <ul style="list-style-type: none"> • OPTIMISTIC—Assumes that Fast InfosetContent is enabled on the service. All requests will be sent using Fast InfosetContent. • PESSIMISTIC—Initial request from client is sent without Fast InfosetContent enabled, but with an HTTP Accept header that indicates that the client supports the Fast InfosetContent capability. If the service response is in Fast InfosetContent format, confirming that Fast InfosetContent is enabled on the service, then subsequent requests from the client will be sent in Fast InfosetContent format. • NONE—Client requests will not use Fast InfosetContent. <p>For more information, see:</p> <ul style="list-style-type: none"> • JAX-WS Web Services: "Configuring the Content Negotiation Strategy" in <i>Developing JAX-WS Web Services for Oracle WebLogic Server</i>. • Oracle Infrastructure Web Services: "Configuring the Content Negotiation Strategy" in <i>Developing Oracle Infrastructure Web Services</i>. 	NONE
<code>enabled</code>	Boolean value that specifies whether or not the feature is enabled.	true

A.3.9.2 @FastInfosetContentCallbackClient Example

```
@FastInfosetContentCallbackClient(
    enable=true,
```



```

        fastInfosetContentNegotiation=FastInfosetContentNegotiationType.OPTIMISTIC
    )

```

A.3.10 @FastInfosetContentClient

The `com.oracle.webservices.api.FastInfosetContentClient` annotation enables and configures Fast InfosetContent on a Web service client.

For more information about Fast InfosetContent, see:

- JAX-WS Web Services: "Optimizing XML Transmission Using Fast InfosetContent" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.
- Oracle Infrastructure Web Services: "Optimizing XML Transmission Using Fast InfosetContent" in *Developing Oracle Infrastructure Web Services*.

A.3.11 @FastInfosetContentService

The `com.oracle.webservices.api.FastInfosetContentService` annotation enables Fast InfosetContent on the web service.

For more information about Fast InfosetContent, see:

- JAX-WS Web Services: "Optimizing XML Transmission Using Fast InfosetContent" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.
- Oracle Infrastructure Web Services: "Optimizing XML Transmission Using Fast InfosetContent" in *Developing Oracle Infrastructure Web Services*.

A.3.11.1 @FastInfosetContentService Attribute

The following table defines the attribute that can be passed to the `com.oracle.webservices.api.FastInfosetContentService` annotation.

Table A-11 Attribute for `com.oracle.webservices.api.FastInfosetContentService` Annotation

Attribute	Description	Default
<code>enabled</code>	Boolean value that specifies whether or not the feature is enabled.	<code>true</code>

A.3.11.2 @FastInfosetContentService Example

```

@FastInfosetContent(
    enable=true
)

```

A.3.12 @JMSTransportClient

The `com.oracle.webservices.api.jms.JMSTransportClient` annotation enables and configures SOAP over JMS transport for Oracle Infrastructure and JAX-WS (Java EE) web service clients.

Using SOAP over JMS transport, web services and clients communicate using JMS destinations instead of HTTP connections, offering the following benefits:

- Reliability

- Scalability
- Quality of service

For more information about using SOAP over JMS transport, see "Using SOAP Over JMS Transport" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.

 **Note:**

SOAP over JMS transport is not compatible with the following web service features: reliable messaging and HTTP transport-specific security.

For Oracle Infrastructure web services, SOAP over JMS transport is not compatible with asynchronous web services.

A.3.12.1 @JMSTransportClient Attributes

For information about the attributes that you can configure using the `@JMSTransportClient` annotation, see "Configuring JMS Transport Properties" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.

A.3.12.2 @JMSTransportClient Example

```
@JMSTransportClient (
    destinationName="myQueue",
    replyToName="myReplyToQueue",
    jndiURL="t3://localhost:7001",
    jndiInitialContextFactory="weblogic.jndi.WLInitialContextFactory",
    jndiConnectionFactoryName="weblogic.jms.ConnectionFactory",
    timeToLive=1000, priority=1,
    messageType=com.oracle.webservices.api.jms.JMSMessageType.TEXT
)
```

A.3.13 @JMSTransportService

The `com.oracle.webservices.api.jms.JMSTransportService` annotation enables and configures SOAP over JMS transport for Oracle Infrastructure and JAX-WS (Java EE) web services.

Using SOAP over JMS transport, web services and clients communicate using JMS destinations instead of HTTP connections, offering the following benefits:

- Reliability
- Scalability
- Quality of service

For more information about using SOAP over JMS transport, see "Using SOAP Over JMS Transport" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.

 **Note:**

SOAP over JMS transport is not compatible with the following web service features: reliable messaging and HTTP transport-specific security.

For Oracle Infrastructure web services, SOAP over JMS transport is not compatible with asynchronous web services.

A.3.13.1 @JMSTransportService Attributes

For information about the attributes that you can configure using the `@JMSTransportService` annotation, see "Configuring JMS Transport Properties" in *Developing JAX-WS Web Services for Oracle WebLogic Server*.

A.3.13.2 @JMSTransportService Example

```
@JMSTransportService(destinationName="myQueue",
    activationConfig = {
        @ActivationConfigProperty(
            propertyName = "destinationType",
            propertyValue = "TOPIC"),
        @ActivationConfigProperty(
            propertyName = "subscriptionDurability",
            propertyValue = "Durable"),
        @ActivationConfigProperty(propertyName = "topicMessagesDistributionMode",
            propertyValue = "One-Copy-Per-Application")})
```

A.3.14 @ManagementPolicy

The `oracle.webservices.annotations.ManagementPolicy` annotation attaches a management policy to the web service.

 **Note:**

This annotation has been deprecated. Oracle recommends that you use the `oracle.wsm.metadata.annotation.PolicyReference` annotation, as described in "[@PolicyReference](#)".

This annotation applies to Oracle Infrastructure web services only.

A.3.14.1 @ManagementPolicy Attributes

The following table defines the attributes that can be passed to the `oracle.webservices.annotations.ManagementPolicy` annotation.

Table A-12 Attributes for `oracle.webservices.annotations.ManagementPolicy` Annotation

Attribute	Description	Default
enabled	Boolean value that specifies whether or not the policy is enabled.	true

Table A-12 (Cont.) Attributes for oracle.webservices.annotations.ManagementPolicy Annotation

Attribute	Description	Default
value	Location from which to retrieve the WS-Policy file. Use the <code>http:</code> prefix to specify the URL of a WS-Policy file on the Web. Use the <code>policy:</code> prefix to specify that the WS-Policy file is packaged in the policy repository.	""

A.3.14.2 @ManagementPolicy Example

```
@ManagementPolicy(
    value="oracle/log_policy",
    enabled = true
)
```

A.3.15 @MaxRequestSize

The `com.oracle.webservices.api.MaxRequestSize` annotation enables you to configure the maximum size, in bytes, of the request message that can be sent to the web service.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.15.1 @MaxRequestSize Attributes

The following table defines the attributes that can be passed to the `com.oracle.webservices.api.MaxRequestSize` annotation.

Table A-13 Attributes for com.oracle.webservices.api.MaxRequestSize Annotation

Attribute	Description	Default
enabled	Boolean value that specifies whether or not the policy is enabled.	true
maxRequestSize	Maximum size of the request message, in bytes. -1 indicates that there is no limit to the size of the message.	-1

A.3.15.2 @MaxRequestSize Example

```
@MaxRequestSize(
    maxRequestSize=-1,
    enabled = true
)
```

A.3.16 @MEXRequestProcessingService

The `com.oracle.webservices.api.MEXRequestProcessingService` annotation enables the exchange of web service metadata.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.16.1 @MEXRequestProcessingService Attribute

The following table defines the attribute that can be passed to the `com.oracle.webservices.api.MEXRequestProcessingService` annotation.

Table A-14 Attribute for `com.oracle.webservices.api.MEXRequestProcessingService` Annotation

Attribute	Description	Default
enabled	Boolean value that specifies whether or not the policy is enabled.	true

A.3.16.2 @MEXRequestProcessingService Example

```
@MEXRequestProcessingService(
    enabled = true
)
```

A.3.17 @MTOM

The `javax.xml.ws.soap.MTOM` annotation enables the use of Message Transmission Optimization Mechanism (MTOM) on the web service. MTOM defines a method for optimizing the transmission of XML data of type `xs:base64Binary` or `xs:hexBinary` in SOAP messages.

For more information, see:

- "Optimizing Binary Data Transmission" in *Developing JAX-WS Web Services for Oracle WebLogic Server*
- @MTOM Javadoc at <http://docs.oracle.com/javase/7/docs/api/javax/xml/ws/soap/MTOM.html>

A.3.17.1 @MTOM Attribute

The following table defines the attribute that can be passed to the `javax.xml.ws.soap.MTOM` annotation.

Table A-15 Attribute for javax.xml.ws.soap.MTOM Annotation

Attribute	Description	Default
enabled	Boolean value that specifies whether or not MTOM is enabled.	true

A.3.17.2 @MTOM Example

```
@MTOM(
    enabled = true
)
```

A.3.18 @MTOMEncodeFaultService

The `com.oracle.webservices.api.MTOMEncodeFaultService` annotation enables the creation of MTOM-enabled SOAP fault messages when MTOM is enabled.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.18.1 @MTOMEncodeFaultService Attribute

The following table defines the attribute that can be passed to the `com.oracle.webservices.api.MTOMEncodeFaultService` annotation.

Table A-16 Attribute for com.oracle.webservices.api.MTOMEncodeFaultService Annotation

Attribute	Description	Default
enabled	Boolean value that specifies whether or not MTOM is enabled.	true

A.3.18.2 @MTOMEncodeFaultService Example

```
@MTOMEncodeFaultService(
    enabled = true
)
```

A.3.19 @MtomPolicy

The `oracle.webservices.annotations.MtomPolicy` annotation attaches an MTOM policy to the web service.



Note:

This annotation has been deprecated. Oracle recommends that you use the `oracle.wsm.metadata.annotation.PolicyReference` annotation, as described in "[@PolicyReference](#)".

This annotation applies to Oracle Infrastructure web services only.

A.3.19.1 @MtomPolicy Attributes

The following table defines the attributes that can be passed to the `oracle.webservices.annotations.MtomPolicy` annotation.

Table A-17 Attributes for `oracle.webservices.annotations.MtomPolicy` Annotation

Attribute	Description	Default
<code>enabled</code>	Boolean value that specifies whether or not the policy is enabled.	<code>true</code>
<code>value</code>	Location from which to retrieve the MTOM policy file. Use the <code>http:</code> prefix to specify the URL of a MTOM policy file on the Web. Use the <code>policy:</code> prefix to specify that the MTOM policy file is packaged in the policy repository.	<code>""</code>

A.3.19.2 @MtomPolicy Example

```
@MtomPolicy(
    value="oracle/wsmtom_policy",
    enabled = true
)
```

A.3.20 @OAuth1 Client Policy

The `OAuth1 Client Policy` annotation attaches a management policy which allows applications to use Twitter API using the statically generated consumer and access tokens.



Note:

This annotation applies to Oracle Infrastructure Web services only.

A.3.21 @Persistence

The `oracle.webservices.annotations.Persistence` annotation configures the secure conversation session persistence mechanism for the web service.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.21.1 @Persistence Attributes

The following table defines the attributes that can be passed to the `oracle.webservices.annotations.Persistence` annotation.

Table A-18 Attributes for `oracle.webservices.annotations.Persistence` Annotation

Attribute	Description	Default
<code>enabled</code>	Boolean value that specifies whether or not the policy is enabled.	<code>true</code>
<code>providerName</code>	Identifies the persistence provider registered in the system. Possible values are: <ul style="list-style-type: none"> <code>oracle:jrf:Memory</code> is the in-memory-based persistence provider. <code>oracle:jrf:Coherence</code> is the integrated Coherence provider. 	<code>oracle:jrf:Coherence</code> when the consumer of the policy (web service or client) is running in a WebLogic Server environment <code>oracle:jrf:Memory</code> when the consumer of the policy is running in a standalone JVM environment

A.3.21.2 @Persistence Example

```
@PortableWebService
@SecurityPolicy("oracle/
wss11_username_token_with_message_protection_wssc_service_policy")
@Persistence(providerName="oracle:jrf:Coherence")

public class TestService {
    .....
}
```


A.3.22 @PolicyReference

The `oracle.wsm.metadata.annotation.PolicyReference` annotation attaches a single policy to a subject, and optionally overrides configuration property values.



Note:

This annotation applies to Oracle Infrastructure web services and RESTful web services.

A.3.22.1 @PolicyReference Attributes

The following table defines the attributes that can be passed to the `oracle.wsm.metadata.annotation.PolicyReference` annotation.

Table A-19 Attributes for `oracle.wsm.metadata.annotation.PolicyReference` Annotation

Attribute	Description	Default
<code>enabled</code>	Boolean value that specifies whether or not the policy is enabled.	<code>true</code>
<code>properties</code>	List of <code>oracle.wsm.metadata.annotation.Property</code> annotations that define configuration override property values.	""
<code>value</code>	Location from which to retrieve the MTOM policy file. Use the <code>http:</code> prefix to specify the URL of a MTOM policy file on the Web. Use the <code>policy:</code> prefix to specify that the MTOM policy file is packaged in the policy repository.	""

A.3.22.2 @PolicyReference Example

```
@PolicyReference(
    value = "oracle/binding_permission_authorization_policy",
    properties = {
        @Property(
            name="resource",
            value="com.sun.jersey.samples.helloworld.resources.MyApplication"),
        @Property(
            name="action",
            value="")
    }
)
```

A.3.23 @PolicySet

The `oracle.wsm.metadata.annotation.PolicySet` annotation defines a set of policy references for the web service, and optionally overrides unscoped configuration property values.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.23.1 @PolicySet Attributes

The following table defines the attributes that can be passed to the `oracle.wsm.metadata.annotation.PolicySet` annotation.

Table A-20 Attributes for `oracle.wsm.metadata.annotation.PolicySet` Annotation

Attribute	Description	Default
<code>properties</code>	List of <code>oracle.wsm.metadata.annotation.Property</code> annotations that define configuration override property values.	""
<code>references</code>	List of <code>oracle.wsm.metadata.annotation.PolicyReference</code> annotations that define policies to attach to the subject.	""

A.3.23.2 @PolicySet Example

```
@PolicySet(references = {
    @PolicyReference("oracle/wss_http_token_service_policy")
})
```

A.3.24 @POXHttpBindingService

The `com.oracle.webservices.api.POXHttpBindingService` annotation enables an endpoint to receive non-SOAP XML messages that are processed by a user defined `javax.xml.ws.Provider<T>.invoke` method.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.24.1 @POXHttpBindingService Attribute

The following table defines the attribute that can be passed to the `com.oracle.webservices.api.POXHttpBindingService` annotation.

Table A-21 Attribute for com.oracle.webservices.api.POXHttpBindingService Annotation

Attribute	Description	Default
enabled	Boolean value that specifies whether or not MTOM is enabled.	true

A.3.24.2 @POXHttpBindingService Example

```
@POXHttpBindingService(
    enabled = true
)
```

A.3.25 @Property

The `oracle.wsm.metadata.annotation.Property` annotation defines a single property that can be used to override the configuration of one or more policies.



Note:

This annotation applies to Oracle Infrastructure web services and RESTful web services.

A.3.25.1 @Property Attributes

The following table defines the attributes that can be passed to the `oracle.wsm.metadata.annotation.Property` annotation.

Table A-22 Attributes for oracle.wsm.metadata.annotation.Property Annotation

Attribute	Description	Default
properties	List of <code>oracle.wsm.metadata.annotation.Property</code> annotations that define unscoped configuration override property values.	""
references	List of <code>oracle.wsm.metadata.annotation.PolicyReference</code> annotations that define policies to attach to the subject.	""

A.3.25.2 @Property Example

```
@PolicyReference(
    value = "oracle/binding_permission_authorization_policy",
    properties = {
        @Property(
            name="resource",
            value="com.sun.jersey.samples.helloworld.resources.MyApplication"),
        @Property(
            name="action",
            value="")
    }
)
```

```
)
}
```

A.3.26 @ReliabilityPolicy

The `oracle.webservices.annotations.ReliabilityPolicy` annotation attaches the `oracle/wsrml0_policy` or `oracle/wsrml0_policy` reliable messaging policies to the web service.



Note:

This annotation has been deprecated. Oracle recommends that you use the `com.oracle.webservices.api.rm.ReliableMessaging` annotation, as described in "[@ReliableMessaging](#)".

This annotation applies to Oracle Infrastructure web services only.

A.3.26.1 @ReliabilityPolicy Attributes

The following table defines the attributes that can be passed to the `oracle.webservices.annotations.ReliabilityPolicy` annotation.

Table A-23 Attributes for `oracle.webservices.annotations.ReliabilityPolicy` Annotation

Attribute	Description	Default
<code>enabled</code>	Boolean value that specifies whether or not the policy is enabled.	<code>true</code>
<code>value</code>	Location from which to retrieve the reliable messaging policy file. Use the <code>http:</code> prefix to specify the URL of a reliable messaging policy file on the Web. Use the <code>policy:</code> prefix to specify that the reliable messaging policy file is packaged in the policy repository.	<code>""</code>

A.3.26.2 @ReliabilityPolicy Example

```
@ReliabilityPolicy(
    value="oracle/wsrml1_policy",
    enabled = true)
```

A.3.27 @ReliableMessaging

The `com.oracle.webservices.api.rm.ReliableMessaging` annotation attaches the `oracle/reliable_messaging_policy` policy to the web service.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.27.1 @ReliableMessaging Attributes

The following table defines the attributes that can be passed to the `oracle.webservices.annotations.ReliableMessaging` annotation.

Table A-24 Attributes for `oracle.webservices.annotations.ReliableMessaging` Annotation

Attribute	Description	Default
<code>enabled</code>	Boolean value that specifies whether or not the policy is enabled.	<code>true</code>
<code>value</code>	Location from which to retrieve the reliable messaging policy file. Use the <code>http:</code> prefix to specify the URL of a reliable messaging policy file on the Web. Use the <code>policy:</code> prefix to specify that the reliable messaging policy file is packaged in the policy repository.	<code>""</code>

A.3.27.2 @ReliableMessaging Example

```
@ReliableMessaging(
    value="oracle/reliable_messaging_policy",
    enabled = true)
```

A.3.28 @RequestProcessingService

The `com.oracle.webservices.api.RequestProcessingService` annotation enables the web service endpoint to process incoming requests.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.28.1 @RequestProcessingService Attribute

The following table defines the attribute that can be passed to the `oracle.webservices.annotations.RequestProcessingService` annotation.

Table A-25 Attribute for `oracle.webservices.annotations.RequestProcessingService` Annotation

Attribute	Description	Default
<code>enabled</code>	Boolean value that specifies whether or not the policy is enabled.	<code>true</code>

A.3.28.2 @RequestProcessingService Example

```
@RequestProcessingService(
    enabled = true)
```

A.3.29 @SchemaValidation

The `com.oracle.webservices.api.SchemaValidation` annotation enables the validation of request messages against the schema.

**Note:**

This annotation applies to Oracle Infrastructure web services only.

A.3.29.1 @SchemaValidation Attribute

The following table defines the attributes that can be passed to the `oracle.webservices.annotations.SchemaValidation` annotation.

Table A-26 Attribute for `oracle.webservices.annotations.SchemaValidation` Annotation

Attribute	Description	Default
<code>enabled</code>	Boolean value that specifies whether or not the policy is enabled.	<code>true</code>

A.3.29.2 @SchemaValidation Example

```
@SchemaValidation(
    enabled = true)
```

A.3.30 @SecurityPolicies (Oracle Infrastructure Web Services)

The `oracle.webservices.annotations.SecurityPolicies` annotation specifies an array of `oracle.webservices.annotations.SecurityPolicy` annotations. Use this annotation if you want to attach more than one WS-Policy file to a class.

**Note:**

This annotation has been deprecated. Oracle recommends that you use the `oracle.wsm.metadata.annotation.PolicySet` annotation, as described in "[@PolicySet](#)".

This annotation applies to Oracle Infrastructure web services only.

For example:

```
@SecurityPolicies({
    @SecurityPolicy(uri=
        "policy:oracle/wss10_username_token_with_message_protection_server_policy"),
    @SecurityPolicy(uri="policy:oracle/authorization_policy")
})
```

A.3.31 @SecurityPolicies (Java EE Web Services)

The `weblogic.wsee.jws.jaxws.owsm.SecurityPolicies` annotation specifies an array of `weblogic.wsee.jws.jaxws.owsm.SecurityPolicy` annotations. Use this annotation if you want to attach more than one WS-Policy file to a class.



Note:

This annotation applies to Java EE web services only.

A.3.31.1 @SecurityPolicies Example

```
@SecurityPolicies({
    @SecurityPolicy(uri=
        "policy:oracle/wss10_username_token_with_message_protection_server_policy"),
    @SecurityPolicy(uri="policy:oracle/authorization_policy")
})
```

A.3.32 @SecurityPolicy (Oracle Infrastructure Web Services)

The `oracle.webservices.annotations.SecurityPolicy` annotation attaches a security policy to the request or response SOAP message. This annotation can be used on its own to apply a single WS-Policy file to a class. If you want to apply more than one WS-Policy file to a class, use the `@SecurityPolicies` annotation to group them together.



Note:

This annotation has been deprecated. Oracle recommends that you use the `oracle.wsm.metadata.annotation.PolicyReference` annotation, as described in "[@PolicyReference](#)".

This annotation applies to Oracle Infrastructure web services only.

A.3.32.1 @SecurityPolicy Attributes

The following table summarizes the attributes that you can pass to the `oracle.webservices.annotations.SecurityPolicy` annotation.

Table A-27 Attributes for `oracle.webservices.annotations.SecurityPolicy` Annotation

Attribute	Description	Default
<code>enabled</code>	Optional. Boolean value that specifies whether the policy is enabled.	<code>true</code>
<code>Properties</code>	Optional. Array of property value-name pairs.	""
<code>value</code>	Location from which to retrieve the WS-Policy file. Use the <code>http:</code> prefix to specify the URL of a WS-Policy file on the Web. Use the <code>policy:</code> prefix to specify that the WS-Policy file is packaged in the policy repository.	""

A.3.32.2 @SecurityPolicy Example

```
@SecurityPolicy(value=
    "policy:oracle/wss10_username_token_with_message_protection_server_policy"),
```

A.3.33 @SecurityPolicy (Java EE Web Services)

The `weblogic.wsee.jws.jaxws.owsm.SecurityPolicy` annotation attaches a security policy to the request or response SOAP message. This annotation can be used on its own to apply a single WS-Policy file to a class. If you want to apply more than one WS-Policy file to a class, use the `@SecurityPolicies` annotation to group them together.



Note:

This annotation applies to Java EE web services only.

A.3.33.1 @SecurityPolicy Attributes

The following table summarizes the attributes that you can pass to the `weblogic.wsee.jws.jaxws.owsm.SecurityPolicy` annotation.

Table A-28 Attributes for `weblogic.wsee.jws.jaxws.owsm.SecurityPolicy` Annotation

Attribute	Description	Default
<code>enabled</code>	Optional. Boolean value that specifies whether the policy is enabled.	<code>true</code>
<code>uri</code>	Location from which to retrieve the WS-Policy file. Use the <code>http:</code> prefix to specify the URL of a WS-Policy file on the Web. Use the <code>policy:</code> prefix to specify that the WS-Policy file is packaged in the policy repository.	<code>""</code>

A.3.33.2 @SecurityPolicy Example

```
@SecurityPolicy(value=
    "policy:oracle/wss10_username_token_with_message_protection_server_policy")
```

A.3.34 @SOAPRequestProcessingService

This `com.oracle.webservices.api.SOAPRequestProcessingService` annotation enables the processing of SOAP requests on a web service endpoint.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.34.1 @SOAPRequestProcessingService Attribute

The following table defines the attribute that can be passed to the `oracle.webservices.annotations.SOAPRequestProcessingService` annotation.

Table A-29 Attribute for `oracle.webservices.annotations.SOAPRequestProcessingService` Annotation

Attribute	Description	Default
<code>enabled</code>	Boolean value that specifies whether or not the policy is enabled.	<code>true</code>

A.3.34.2 @SOAPRequestProcessingService Example

```
@SOAPRequestProcessingService(
    enabled = true)
```

A.3.35 @TestPageProcessingService

The `com.oracle.webservices.api.TestPageProcessingService` annotation enables the Web Service Test Client. This annotation applies to Oracle Infrastructure web services only.

For more information, see "Using the Web Services Test Client" in *Administering Web Services*.

 **Note:**

The procedures described in this section do not impact the availability of the **Web Services Test** link on the Web Service Endpoint page, which enables you to access the Fusion Middleware Control Test Web Service page. For more information, see "Using the Test Web Service Page in Fusion Middleware Control" in *Administering Web Services*.

A.3.35.1 @TestPageProcessingService Attribute

The following table defines the attribute that can be passed to the `oracle.webservices.annotations.TestPageProcessingService` annotation.

Table A-30 Attribute for `oracle.webservices.annotations.TestPageProcessingService` Annotation

Attribute	Description	Default
<code>enabled</code>	Boolean value that specifies whether or not the policy is enabled.	<code>true</code>

A.3.35.2 @TestPageProcessingService Example

```
@TestPageProcessingService(
    enabled = true)
```

A.3.36 @WSDLRequestProcessingService

This `com.oracle.webservices.api.WSDLRequestProcessingService` annotation enables access to the WSDL for the web service.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.36.1 @WSDLRequestProcessingService Attribute

The following table defines the attribute that can be passed to the `oracle.webservices.annotations.WSDLRequestProcessingService` annotation.

Table A-31 Attribute for `oracle.webservices.annotations.WSDLRequestProcessingService` Annotation

Attribute	Description	Default
enabled	Boolean value that specifies whether or not the policy is enabled.	true

A.3.36.2 @WSDLRequestProcessingService Example

```
@WSDLRequestProcessingService(
    enabled = true)
```

A.3.37 @WSLoggingLevel

This `com.oracle.webservices.api.WSLoggingLevel` annotation sets the logging level for diagnostic logs for the web service endpoint.



Note:

This annotation applies to Oracle Infrastructure web services only.

A.3.37.1 @WSLoggingLevel Attributes

The following table defines the attributes that can be passed to the `oracle.webservices.annotations.WSLoggingLevel` annotation.

Table A-32 Attribute for `oracle.webservices.annotations.WSLoggingLevel` Annotation

Attribute	Description	Default
enabled	Boolean value that specifies whether or not the policy is enabled.	true

Table A-32 (Cont.) Attribute for oracle.webservices.annotations.WSLoggingLevel Annotation

Attribute	Description	Default
loggingLevel	Defines the logging level. Valid values include: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, or NULL.	""

A.3.37.2 @WSLoggingLevel Example

```
@WSLoggingLevel(
    enabled = true,
    loggingLevel=INFO
)
```

B

Predefined Assertion Templates for Oracle Web Services

Use the assertion template settings and configuration properties for Oracle web services to customize the assertion template.

Topics:

- [Assertion Template Settings for Oracle Web Services](#)
- [Assertion Template Configuration Properties for Oracle Web Services](#)

B.1 Assertion Template Settings for Oracle Web Services

Use the assertion template setting to set the predefined assertion template.

The settings are listed alphabetically.



Note:

Not all settings apply to all assertion templates.

- [Action Match](#)
- [Algorithm Suite](#)
- [Authentication Header—Header Name](#)
- [Authentication Header—Mechanism](#)
- [Body Elements](#)
- [Bootstrap Message Security](#)
- [Client Entropy](#)
- [Client Policy URI](#)
- [Confirm Signature](#)
- [Confirmation Type](#)
- [Constraint Match](#)
- [Creation Time Required](#)
- [Derived Keys](#)
- [Enabled](#)
- [Encrypt Signature](#)
- [Encryption Key Reference Mechanism](#)
- [Fault](#)
- [Fault Message Settings](#)

- Header Elements
- Include Entire Body
- Include MIME Headers
- Include SwA Attachment
- Include Timestamp
- Is Encrypted
- Is Signed
- Kerberos Token Type
- Key Type
- Keystore Recipient Alias
- Mutual Authentication Required
- Name Identifier Format
- Nonce Required
- Password Type
- Permissions
- Permission Class
- Port Endpoint
- Port URI
- Re-authenticate
- Recipient Encryption Key Reference Mechanism
- Recipient Sign Key Reference Mechanism
- Request
- Request Message Settings
- Request XPath
- Request Namespaces
- Require Applies To
- Require Client Entropy
- Require External Reference
- Require Internal Reference
- Require Server Entropy
- Resource Match
- Response
- Response Message Settings
- Response Namespaces
- Response XPath
- Roles
- Server Entropy
- Sign Key Reference Mechanism

- [Sign Then Encrypt](#)
- [Token Type](#)
- [Transport Layer Security](#)
- [Transport Layer Security—Include Timestamp](#)
- [Transport Layer Security—Mutual Authentication Required](#)
- [Version](#)
- [Trust Version](#)
- [Use Derived Keys](#)
- [Use PKI Path](#)
- [WSDL Exist](#)
- [WSDL](#)

B.1.1 Action Match

Action Match performs authorization checks for web service operation.

This value can be a comma-separated list of values. This field accepts wildcards. For example, `validate,amountAvailable`.

B.1.2 Algorithm Suite

Algorithm suite is used for message protection.

For more information, see "[Supported Algorithm Suites](#)".

B.1.3 Authentication Header—Header Name

Specifies the name of the authentication header.

B.1.4 Authentication Header—Mechanism

This setting is used to specify the authentication mechanism for the assertion template.

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 "[About Configuring Keystores for SSL](#)".
- `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`—Reserved for future use.
- `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.

B.1.5 Body Elements

This setting is used to define the body elements for the assertion template.

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**.

B.1.6 Bootstrap Message Security

A Secure Conversation policy has two policies: inner and outer. The Bootstrap Message Security control exposes the inner and outer policies.

The bootstrap (inner) policy is used to obtain the token and establish the handshake between the client and the web service. The outer policy is used for application messages when making requests with the token.

B.1.7 Client Entropy

This is used as key material for the requested proof token in Secure Conversation.

B.1.8 Client Policy 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. In some cases, you can filter the list of policies by selecting either **Show All Client Policies** or **Show Compatible Client Policies**. If you choose **Show Compatible Client Policies**, only those policies compatible with the port specified in [Port URI](#) are shown.

B.1.9 Confirm Signature

This flag specifies whether to send a signature confirmation back to the client. The default value of this flag is 'false'.

B.1.10 Confirmation Type

This setting specifies the Sender Vouches SAML token for authentication.

The only valid value is:

- sender-vouches—Uses the Sender Vouches SAML token for authentication.

B.1.11 Constraint Match

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 !.

B.1.12 Creation Time Required

This flag 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.

B.1.13 Derived Keys

This flag specifies whether derived keys should be used.

B.1.14 Enabled

For the preconfigured WS-SC policies, Secure Conversation is enabled by default. For all of the other policies, Secure Conversation is disabled by default.

B.1.15 Encrypt Signature

This flag specifies whether to encrypt the signature.

B.1.16 Encryption Key Reference Mechanism

Mechanism used when encrypting the request.

Valid values for `wss10_message_protection_client_template` and `wss10_saml_token_with_message_protection_client_template`:

- `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.

Valid values for `wss11_message_protection_client_template`, `wss11_saml_token_with_message_protection_client_template`, `wss11_saml20_token_with_message_protection_client_template`, `wss11_username_token_with_message_protection_client_template`, `wss11_x509_token_with_message_protection_client_template`, and `wss11_username_token_with_message_protection_client_template`:

- `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.

B.1.17 Fault

Use this setting for logging fault 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.

B.1.18 Fault Message Settings

Use these settings for message signing and encryption.

See [Table 18-131](#).

B.1.19 Header Elements

Use this setting to 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**.

B.1.20 Include Entire Body

Use this setting to sign or encrypt the entire body of the SOAP message.

If false, you can add specific body elements using the Body Elements section.

B.1.21 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.

B.1.22 Include SwA Attachment

Include SwA Attachment sign or encrypt SOAP messages with attachments.

Note: This field is not applicable to MTOM attachments.

B.1.23 Include Timestamp

This flag 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.

B.1.24 Is Encrypted

This flag specifies whether the SAML token is encrypted.

B.1.25 Is Signed

This flag specifies whether the SAML token is signed.

B.1.26 Kerberos Token Type

This specifies the type of Kerberos token.

The only valid value is: gss-apreq-v5 (Kerberos Version 5 GSS-API).

B.1.27 Key Type

This specifies the key type.

The only valid value is: bearer.

B.1.28 Keystore Recipient Alias

The alias of the STS certificate you added to the keystore.

The default alias name is sts-csf-key.

B.1.29 Mutual Authentication Required

This flag 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.

B.1.30 Name Identifier Format

This specifies the type of format to be used for the name identifier.

Specify one of the following values:

- unspecified
- emailAddress
- X509SubjectName
- WindowsDomainQualifiedName

The following assertion templates have the additional value: kerberos:

wss10_saml20_token_client_template, wss_saml20_token_bearer_over_ssl_client_template,
wss10_saml20_token_with_message_protection_client_template,
wss11_saml20_token_with_message_protection_client_template

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 "[Configuring SAML Web Service Client at Design Time](#)"). 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.

B.1.31 Nonce Required

This flag 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.

B.1.32 Password Type

This specifies the type of password required.

Valid values are:

- `none`—No password.
- `plaintext`—Password in clear text.
- `digest`— Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest.

If you specify a password type of `None`, you do not need to include a password in the key.

Note:

If you do not use a digest password, policies created using this template are not secure; `plaintext` transmits the password in clear text. You should use this assertion without a digest password 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](#)".

B.1.33 Permissions

Role- and permission-based policies use the guard element 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.

For more information on guard element, see "[orawsp:guard Element](#)".

By default, resource name and action use the wildcard asterisk "*" and everything is allowed.

B.1.34 Permission Class

Class used for the permission-based checking.

For example, `oracle.wsm.security.WSFuncPermission`.

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://docs.oracle.com/javase/7/docs/api/java/security/Permission.html>.

The default is `oracle.wsm.security.WSFunctionPermission`.

B.1.35 Port Endpoint

The endpoint of the STS web service. For a WSDL 2.0 STS, the format is specified as `targetnamespace#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)`.

B.1.36 Port URI

The actual endpoint URI of the STS port.

For example, `http://host:port/context-root/service1`.

B.1.37 Re-authenticate

You can enable the re-authenticate control only for SAML sender vouches policies when the `propagate.identity.context` configuration attribute is set to `True`.

For more information, see "When to Use Re-Authentication" in *Understanding Oracle Web Services Manager*.

B.1.38 Recipient Encryption Key Reference Mechanism

Mechanism used when encrypting the receipt.

Valid values are the same as for Sign Key Reference Mechanism above.

B.1.39 Recipient Sign Key Reference Mechanism

Mechanism used when signing the receipt.

Valid values are the same as for "Sign Key Reference Mechanism".

B.1.40 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.

B.1.41 Request Message Settings

Specifies the request message settings.

See [Table 18-131](#).

B.1.42 Request XPath

This specifies if XPath should be requested.

Optional element. A comma-separated list of XPath for the request. Default value is blank.

B.1.43 Request Namespaces

Optional element. A comma-separated list of namespaces for the request, where each namespace has a prefix and URI separated by the equals sign. Default value is blank.

B.1.44 Require Applies To

Optional element in the RST. If present, OWSM 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.

B.1.45 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.

B.1.46 Require External Reference

Indicates whether external reference to the token is required.

B.1.47 Require Internal Reference

Indicates whether internal reference to the token is required.

B.1.48 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.

B.1.49 Resource Match

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`.

B.1.50 Response

Requirements for logging response messages. The valid values are the same as for [Request](#).

B.1.51 Response Message Settings

See [Table 18-131](#).

B.1.52 Response Namespaces

Optional element. A comma-separated list of namespaces, where each namespace has a prefix and URI separated by the equals sign. Default value is blank.

B.1.53 Response XPath

Optional element. A comma-separated list of XPath expressions for the response. Default value is blank.

B.1.54 Roles

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**.

B.1.55 Server Entropy

This is used as key material for the requested proof token for Secure Conversation.

B.1.56 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 property is valid only for the following templates: `wss11_saml_token_with_message_protection_client_template`, `wss11_saml20_token_with_message_protection_client_template`, `wss11_x509_token_with_message_protection_client_template`, `wss11_sts_issued_saml_with_message_protection_client_template`, `oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template`.

B.1.57 Sign Then Encrypt

This flag specifies whether the request is signed and then encrypted.

B.1.58 Token Type

SAML token type. The only valid value is: `1.1`.

B.1.59 Transport Layer Security

This flag specifies whether Secure Socket Layer (SSL), otherwise known as Transport Layer Security (TLS), is enabled.

B.1.60 Transport Layer Security—Include Timestamp

This flag 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.

B.1.61 Transport Layer Security—Mutual Authentication Required

This flag 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.

B.1.62 Version

This specifies the SAML or Secure Conversation version.

B.1.63 Trust Version

This specifies the WS-Trust version.

B.1.64 Use Derived Keys

This flag specifies whether derived keys should be used.

B.1.65 Use PKI Path

This flag specifies whether X509PKIPathV1 tokens should be processed and propagated.

B.1.66 WSDL Exist

Indicates whether a WSDL exists for the Security Token Service (STS).

If a WSDL does exist, you may be prompted when cloning the template to enter the endpoint URI for the WSDL, indicate whether authentication is required, and enter a user name and password. You can then select **Parse WSDL** to parse the WSDL and fill the subsequent fields with values from the WSDL.

B.1.67 WSDL

The endpoint URI of an STS WSDL, used to obtain STS information and invoke the STS for token exchange.

B.2 Assertion Template Configuration Properties for Oracle Web Services

The following sections summarize the configuration properties that can be set for the predefined assertion templates; settings are listed alphabetically.



Note:

Not all configuration properties apply to all assertion templates.

- algorithm
- anonymous.access
- application.name
- attesting.mapping.attribute
- caller.principal.name
- credential.delegation
- csf.map
- csf-key
- encryption-algorithm
- execute.action
- ignore.timestamp.in.response
- include-timestamp
- issued.token.caching
- issued.token.lifetime
- iteration
- keysize
- keytab.location
- keystore.enc.csf.key
- keystore.recipient.alias
- keystore.sig.csf.key
- lookup.action
- on.behalf.of
- policy.reference.uri
- port.endpoint
- port.uri
- propagate.identity.context
- realm

- `reference.priority`
- `resource.mapping.model`
- `resource.name`
- `resource.type`
- `rm.encrypt.body`
- `role`
- `salt`
- `saml.assertion.filename`
- `saml.audience.uri`
- `saml.envelope.signature.required`
- `saml.issuer.name`
- `saml.trusted.issuers`
- `sc.token.lifetime`
- `service.principal.name`
- `subject.precedence`
- `sts.auth.caller.principal.name`
- `sts.auth.keytab.location`
- `sts.auth.on.behalf.of.csf.key`
- `sts.auth.on.behalf.of.username.only`
- `sts.auth.service.principal.name`
- `sts.auth.user.csf.key`
- `sts.auth.x509.csf.key`
- `sts.in.order`
- `sts.keystore.recipient.alias`
- `use.single.step`
- `user.attributes`
- `user.roles.include`
- `user.tenant.name`
- `wSDL.uri`
- `auth.header.token.type`

B.2.1 algorithm

The key derivation algorithm, which must be PBKDF2.

B.2.2 anonymous.access

A configuration override property with default value **true**. Value can be **true** or **false**. When set to true, it governs the addition of anonymous subject in the message context.

B.2.3 application.name

The application name defined in OES. Value can be static or dynamic that uses `${}` notation.

B.2.4 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.

B.2.5 auth.header.token.type

By default, at client side, "oit" authorization header is built as part of send request phase. To send the "Bearer" authorization header from the client side, the client need to override the configuration property "auth.header.token.type" with the value "Bearer". The service side processes the authorization header with "Bearer" as well as "oit". It checks the authorization header for "oit" as well as for "Bearer" as auth header prefix. If it matches, the service side extracts the header and verifies it, otherwise throws an appropriate exception.

B.2.6 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.

B.2.7 credential.delegation

Flag that specifies whether Credential Delegation with Forwarded TGT is supported. For more information, see "[About Configuration of Credential Delegation](#)". This value is false by default.

B.2.8 csf.map

Oracle WSM map in the credential store that contains the CSF aliases.

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 "[About Creating an Application-level Credential Map](#)".

B.2.9 csf-key

Credential Store Key that maps to a username and password in the OPSS identity store. For information about how to add the key to the credential store, see "[Adding Keys and User Credentials to Configure the Credential Store](#)".

B.2.10 encryption-algorithm

The data encryption algorithm. It must be `AES/CBC/PKCS5Padding`.

B.2.11 execute.action

Optional property. Action that will be used during real authorization. Value can be static or dynamic that uses `${}` notation.

B.2.12 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.

B.2.13 include-timestamp

It validates the timestamp and responds back with timestamp in the WS-Security Header Element.

This default value is `false`. If set to `true` then the client will send the timestamp in the WS-Security Header Element. The service then validates the timestamp and responds back with timestamp in the WS-Security Header Element and this will be validated by the client.

B.2.14 issued.token.caching

The issued tokens are cached by OWSM. When making a request to STS, OWSM requests a token lifetime for returned tokens for the period specified by `issued.token.lifetime`.

If the STS returns a token lifetime value different from the requested `issued.token.lifetime` value, OWSM uses the return value as the period for caching returned tokens. If the STS returns an empty token lifetime value, OWSM does not cache returned tokens.

B.2.15 issued.token.lifetime

The time in milliseconds for OWSM to request as the token lifetime when obtaining an issued token from a security token service (STS). The domain default for this value is 28800000 milliseconds (eight hours). For information about how to change this default value, see "[Configuring the Lifetime for the Issued Token Using Fusion Middleware Control](#)".

B.2.16 iteration

The iteration count for key derivation.

B.2.17 iterations

The iteration count for key derivation using password. The default value is 1000. If an invalid iteration count is passed, that is, non-integer parsable string or negative value, a warning message is displayed and the default value of 1000 is used.

B.2.18 keysize

The size of the key for key derivation.

B.2.19 keytab.location

Location of the client's keytab file.

B.2.20 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 `keystore.enc.csf.key`, as described in "[Overview of Policy Configuration Overrides](#)".

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.

B.2.21 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. For information about overriding policies, see "[Overview of Policy Configuration Overrides](#)".

B.2.22 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.

B.2.23 lookup.action

Optional property. Action that will be used during attributes lookup. Value can be static or dynamic that uses `#{}` notation.

B.2.24 on.behalf.of

Optional property. Override this property to indicate whether the request is on behalf of 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 sent in the `onBehalfOf` token. If the `sts.auth.on.behalf.of.username.only` property is also set to true, the password portion of the identity in the CSF key will not be sent in the `onBehalfOf` token.

Otherwise, if the subject is already established, then the username from the subject will be sent as the `onBehalfOf` token.

If `sts.auth.on.behalf.of.csrf.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.

B.2.25 `policy.reference.uri`

It 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.

In some cases, you can filter the list of policies by selecting either **Show All Client Policies** or **Show Compatible Client Policies**. If you choose **Show Compatible Client Policies**, only those policies compatible with the port specified in [Port URI](#) are shown.

B.2.26 `port.endpoint`

The endpoint of the STS web service. For a WSDL 2.0 STS, the format is specified as `targetnamespace#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)`.

B.2.27 `port.uri`

The actual endpoint URI of the STS port. For example, `http://host:port/context-root/service1`.

B.2.28 `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. For more information, see "[Propagating Identity Context Using SAML Policies](#)".

B.2.29 `realm`

HTTP Realm.

B.2.30 `reference.priority`

Note:

This property has no effect when defined as an unscoped override using the `setWSMPolicySetOverride` command. For more information, see "setWSMPolicySetOverride" in *WLST Command Reference for Infrastructure Components*.

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](#)".

B.2.31 remote-user

Optional property that asserts the user and creates subject for Webgate/OAM protected resources. Default value is `OAM_REMOTE_USER`. If the value is set to `NONE`, the support for remote user header is disabled. If `OAM_REMOTE_USER` is present along with other security headers in a request, `OAM_REMOTE_USER` header is given highest priority.

B.2.32 resource.mapping.model

Optional property that switches between different out-of-the-box mapping models. The default value is `operation_as_action`. Other allowed values are `operation_as_resource_hierarchy` and `lookup_action_fixed_execute_action_as_operation`.

B.2.33 resource.name

Optional property. Resource name defined in OES. Value can be static or dynamic that uses `{}` notation.

B.2.34 resource.type

Optional property. Resource type defined in OES. Value can be static or dynamic that uses `${}` notation.

B.2.35 rm.encrypt.body

Applies to web service client only. If this is set, the body of protocol request messages such as `createSequence()` and `terminateSequence()` are encrypted. The default is that WS-RM protocol messages are not encrypted.

The response message body for protocol messages depends on the request message body: if the request message from the client is encrypted for protocol messages, the web service sends the response encrypted, and vice versa.

B.2.36 role

SOAP role.

B.2.37 salt

A non-null and non-empty salt for key derivation.

B.2.38 saml.assertion.filename

Name of the of the SAML token file.

B.2.39 saml.audience.uri

The saml.audience.uri configuration property 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.
- Base URL of the service URL.

B.2.40 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.

B.2.41 saml.issuer.name

SAML issuer URI. For more information, see "[Adding an Additional SAML Assertion Issuer Name](#)".

B.2.42 saml.trusted.issuers

A comma-separated list of SAML token trusted issuers for an application that will override trusted issuers at domain level.

B.2.43 sc.token.lifetime

Secure Conversation token lifetime in milliseconds. The security context is shared by the client and web service for the lifetime of a communication session. This is the time after which the SCT is expired.

B.2.44 service.principal.name

Kerberos principal name that identifies the service.

B.2.45 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.

B.2.46 `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>`.

B.2.47 `sts.auth.keytab.location`

Location of the client's keytab file.

B.2.48 `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). For information about the on behalf of entity, see "[on.behalf.of](#)".

B.2.49 `sts.auth.on.behalf.of.username.only`

Optional property. Use to configure the on behalf of entity when `sts.auth.on.behalf.of.csf.key` is specified. For information about the on behalf of entity, see "[on.behalf.of](#)".

B.2.50 `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@MYREALM.COM`.

B.2.51 `sts.auth.user.csf.key`

Use to configure username/password to authenticate to the STS.

If `policy-reference-uri` in the `oracle/sts_trust_config_template` client assertion template 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.

B.2.52 `sts.auth.x509.csf.key`

Use to configure X509 certificate for authenticating to the STS.

If `policy-reference-uri` in the `oracle/sts_trust_config_template` client assertion template 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.

B.2.53 `sts.in.order`

Use in Web Services Federation cases to specify the STSes in the trust chain from the RP-STS that web service trusts back to the IP-STS that the web client uses to authenticate.

Set the value of `sts.in.order` to a comma separated list of the STS URIs to be contacted, starting with the RP-STS and ending with the IP-STS.

For more information about using this property, see "[About Configuring Web Services Federation](#)".

B.2.54 sts.keystore.recipient.alias

The alias of the STS certificate you added to the keystore. The default alias name is sts-csf-key.

B.2.55 user.csf.key

User Credential Store Key that maps to a username and password in the OPSS identity store. See "[Adding Keys and User Credentials to Configure the Credential Store](#)".

The default value of `basic.credentials` contains the password details of a user. The password details are required to derive key for encryption or signature. The service creates a user csf key for each user.

If the username in the user csf key is different from the one coming in the request header, the authentication fails.

If the password is different, then signature verification or decryption fails.

B.2.56 use.single.step

This is an optional configuration property. Set value to `true` to skip lookup phase.

Does not apply to masking policy.

B.2.57 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 OWSM 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 "[Supported Authentication Providers in WebLogic Server](#)".

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](#)" for more information.

B.2.58 user.roles.include

This configuration property specifies the user roles.

When set to true, OWSM reads the roles of the user from the user repository (LDAP) and propagates them as SAML attributes.

B.2.59 user.tenant.name

This configuration property is reserved for use with Oracle Cloud.

B.2.60 wsdl.uri

The endpoint URI of an STS WSDL, used to obtain STS information and invoke the STS for token exchange.

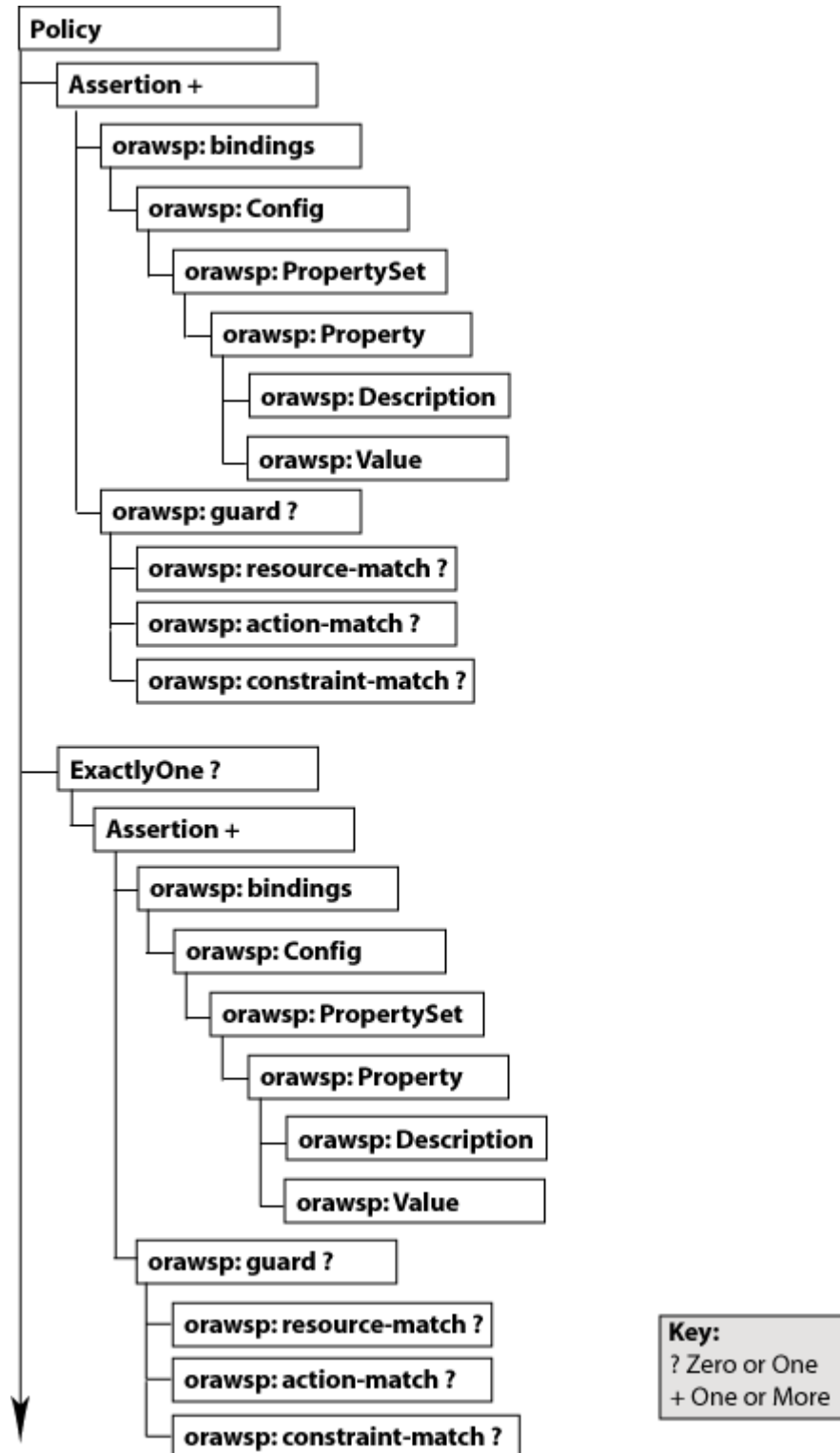
C

Schema Reference for Predefined Assertions for Oracle Web Services

This appendix provides the XML schema for reference when creating a WS-Policy file that contains web service assertions.

The following figure describes the element hierarchy of the assertions in the WS-Policy file.

Figure C-1 Element Hierarchy of an Assertion



This section describes the elements and subelements for the following assertions:

- [wsp:Policy Element](#)
- [wsp:ExactlyOne Element](#)

- orasp:Assertion Element
- orawsp:bindings Element
- orawsp:Config Element
- orawsp:PropertySet Element
- orawsp:Property Element
- orawsp:Description Element
- orawsp:Value Element
- orawsp:guard Element
- orawsp:resource-match Element
- orawsp:action-match Element
- orawsp:constraint-match Element
- oralgp:Logging Element
- orasp:binding-authorization Element
- orasp:binding-permission-authorization Element
- orasp:coreid-security Element
- orasp:http-security Element
- orasp:kerberos-security Element
- orasp:sca-component-authorization Element
- orasp:sca-component-permission-authorization Element
- orasp:sts-trust-config Element
- orasp:wss10-anonymous-with-certificates Element
- orasp:wss10-mutual-auth-with-certificates Element
- orasp:wss10-saml-hok-with-certificates Element
- orasp:wss10-saml-token Element
- orasp:wss10-saml-with-certificates Element
- orasp:wss10-username-with-certificates Element
- orasp:wss11-anonymous-with-certificates Element
- orasp:wss11-mutual-auth-with-certificates Element
- orasp:wss11-saml-with-certificates Element
- orasp:wss11-sts-issued-token-with-certificates Element
- orasp:wss11-username-with-certificates Element
- orasp:wss-saml-token-bearer-over-ssl Element
- orasp:wss-saml-token-over-ssl Element
- orasp:wss-sts-issued-token-over-ssl Element
- orasp:wss-username-token Element
- orasp:wss-username-token-over-ssl Element
- rm:RMAssertion Element
- wsaw:UsingAddressing Element

- wsoma:OptimizedMimeSerialization Element
- oralgp:fault Element
- oralgp:request Element
- oralgp:response Element
- oralgp:msg-log Element
- orasp:attachment Element
- orasp:auth-header Element
- orasp:body Element
- orasp:check-permission Element
- orasp:coreid-token Element
- orasp:denyAll Element
- orasp:element Element
- orasp:encrypted-elements Element
- orasp:encrypted-parts Element
- orasp:fault Element
- orasp:header Element
- orasp:issued-token Element
- orasp:kerberos-token Element
- orasp:msg-security Element
- orasp:permitAll Element
- orasp:request Element
- orasp:require-tls Element
- orasp:response Element
- orasp:role Element
- orasp:saml-token Element
- orasp:signed-elements Element
- orasp:signed-parts Element
- orasp:username-token Element
- orasp:x509-token Element
- orawsp:Description Element

C.1 wsp:Policy Element

This element groups nested policy assertions.

C.1.1 WS-Policy Attributes

The following table summarizes the WS-Policy attributes, including the Oracle extensions.

Table C-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, wsm, addressing, and management.
description	Description of the policy.
displayName	Name displayed in the user interface.
localOptimization	Flag that specifies whether local optimization is enabled. OWSM 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: <ul style="list-style-type: none"> 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.
oraSmartDigest	Smart Digest.
readOnly	Indicates whether clients should be prevented from modifying this policy.
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 created.
id	Policy ID.

C.1.2 Example of WS-Policy

This example shows the code snippet for WS-Policy:

```
<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:wssu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
  utility-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:readOnly="true"
  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>
```

C.2 wsp:ExactlyOne Element

This is an optional element that defines an OR group.

For more information about OR groups, see "Defining Multiple Policy Alternatives (OR Groups)" in About Defining Multiple Policy Alternatives (OR Groups).

C.2.1 wsp:ExactlyOne Element Attribute

The following table summarizes the <wsp:ExactlyOne> element attribute.

Table C-2 Attribute of <wsp:ExactlyOne> Element

Attribute	Description
Name	Set to OR to indicate that this is an OR group.

C.2.2 Example of wsp:ExactlyOne Element

This example shows the code snippet for wsp:ExactlyOne element:

```
<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>
```

C.3 orasp:Assertion Element

orasp:Assertion element is the main element of the assertion.

Valid assertion elements include:

- [oralgp:Logging Element](#)
- [orasp:binding-authorization Element](#)
- [orasp:binding-permission-authorization Element](#)
- [orasp:coreid-security Element](#)
- [orasp:http-security Element](#)
- [orasp:kerberos-security Element](#)
- [orasp:sca-component-authorization Element](#)
- [orasp:sca-component-permission-authorization Element](#)
- [orasp:sts-trust-config Element](#)
- [orasp:wss10-anonymous-with-certificates Element](#)
- [orasp:wss10-mutual-auth-with-certificates Element](#)
- [orasp:wss10-saml-hok-with-certificates Element](#)
- [orasp:wss10-saml-token Element](#)
- [orasp:wss10-saml-with-certificates Element](#)
- [orasp:wss10-username-with-certificates Element](#)
- [orasp:wss11-anonymous-with-certificates Element](#)
- [orasp:wss11-mutual-auth-with-certificates Element](#)
- [orasp:wss11-saml-with-certificates Element](#)
- [orasp:wss11-sts-issued-token-with-certificates Element](#)
- [orasp:wss11-username-with-certificates Element](#)
- [orasp:wss-saml-token-bearer-over-ssl Element](#)
- [orasp:wss-saml-token-over-ssl Element](#)
- [orasp:wss-sts-issued-token-over-ssl Element](#)
- [orasp:wss-username-token Element](#)
- [orasp:wss-username-token-over-ssl Element](#)
- [rm:RMAssertion Element](#)
- [wsaw:UsingAddressing Element](#)
- [wsoma:OptimizedMimeSerialization Element](#)

C.3.1 orasp:Assertion Element Attributes

The following table summarizes the <orasp:Assertion> element attributes.

Table C-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, wsrn, addressing, and management.

C.3.2 Example of orasp:Assertion Element

This example shows the code snippet for orasp:Assertion element:

```
<orasp:wss11-mutual-auth-with-certificates orasp:Enforced="true"
  orasp:Silent="false" orasp:category="security/authentication,
  security/msg-protection"
  orasp:name="WS-Security 1.1 Mutual Auth with certificates">
  ...
</orasp:wss11-mutual-auth-with-certificates>
```

C.4 orawsp:bindings Element

The <orawsp:bindings> element defines the bindings in the assertion.

This element includes the [orawsp:Config Element](#).

C.4.1 Example of orawsp:bindings Element

This example shows the code snippet for orawsp:bindings element:

```
<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>
```

C.5 orawsp:Config Element

The <orawsp:Config> element defines the configuration for the assertion.

This element can include the [orawsp:PropertySet Element](#).

C.5.1 orawsp:Config Element Attributes

The following table summarizes the <orawsp:Config> element attributes.

Table C-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. <ul style="list-style-type: none"> declarative—Use deployment descriptors and configuration files to describe authentication and authorization requirements. programmatic—Embed security enforcement within the application.

C.5.2 Example of orawsp:Config Element

This example shows the code snippet for orawsp:Config element:

```
<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>
```

C.6 orawsp:PropertySet Element

The <orawsp:PropertySet> element groups nested properties.

This element includes the [orawsp:Property Element](#).

C.6.1 orawsp:PropertySet Element Attributes

The following table summarizes the attributes of the <orawsp:PropertySet> element.

Table C-5 Attributes of <orawsp:PropertySet> Element

Attribute	Description
name	Name of the property set.

C.6.2 Example of orawsp:PropertySet Element

```
<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>
```

C.7 orawsp:Property Element

The <orawsp: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 Element](#)

C.7.1 orawsp:Property Element Attributes

The following table summarizes the attributes of the <orawsp:Property> element.

Table C-6 Attributes of <orawsp:Property> Element

Attribute	Description
name	Name of the property. See Table C-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: <ul style="list-style-type: none"> • 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 " Overview of Policy Configuration Overrides ".

The following table summarizes the properties used by the predefined assertions.

Table C-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, <code>validate,amountAvailable</code> .
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.

Table C-7 (Cont.) Properties Used by the Predefined Assertions

Property	Description
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.
credential.delegation	Flag that specifies whether Credential Delegation with Forwarded TGT is supported. For more information, see " About Configuration of Credential Delegation ". This value defaults to false.
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. The default value is <code>basic.credentials</code> .
DeliveryAssurance	Delivery assurance. Valid values include: <ul style="list-style-type: none"> • <code>InOrder</code>—Messages are delivered in the order they were sent. This is the default. • <code>AtLeastOnce</code>—Every message is delivered at least once. It is possible that some messages are delivered more than once. • <code>AtLeastOnceInOrder</code>—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. • <code>ExactlyOnce</code>—Every message is delivered exactly once, without duplication. • <code>ExactlyOnceInOrder</code>—Every message is delivered exactly once, without duplication, and in the order they were sent. • <code>AtMostOnce</code>—Messages are delivered at most once, without duplication. It is possible that some messages may not be delivered at all. • <code>AtMostOnceInOrder</code>—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 <code>StoreType</code> is set to <code>JDBC</code> . This value defaults to <code>jdbc/MessageStore</code> .
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.
keystore.enc.csf.key	If you set this value you then can override <code>keystore.enc.csf.key</code> , as described in " Overview of Policy Configuration Overrides ".
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 " Understanding Service Identity Certificate Extensions ".

Table C-7 (Cont.) Properties Used by the Predefined Assertions

Property	Description
on.behalf.of	Override this property to indicate whether the request is on behalf of another entity. The default value for this flag is false.
permission-class	Class used for the permission-based checking. For example, <code>oracle.wsm.security.WSFuncPermission</code> .
realm	HTTP realm. This value defaults to <code>owsm</code> .
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 <code>http://project11</code> and the service name is <code>CreditValidation</code> , the resource name is <code>http://project11/CreditValidation</code> .
role	SOAP role. This value defaults to <code>ultimateReceiver</code> .
saml.assertion.filename	File containing SAML assertions. This value defaults to <code>temp</code> .
saml.audience.uri	Represents the relying party, as a comma-separated URI. This field accepts the following wildcards: <ul style="list-style-type: none"> • <code>*</code> in any location. • <code>/*</code> at the end of the URI. • <code>.*</code> at the end of the URI.
saml.issuer.name	Name of the issuer of the SAML token. This value defaults to <code>www.oracle.com</code> .
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 <code>oracle</code> .
StoreType	Type of message store. Valid values include: <ul style="list-style-type: none"> • <code>InMemory</code>—Messages are stored in memory. This is the default. • <code>JDBC</code>—Messages are stored using JDBC.
sts.auth.caller.principal.name	Client's principal name as generated using the <code>ktpass</code> command and mapped to the username for which the kerberos token should be generated. It is of the format <code><username>@<REALM NAME></code> .
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 <code><host>/<machine name>@<REALM NAME></code> . For example, <code>HTTP/mymachine@MYREALM.COM</code> .
sts.auth.user.csf.key	Use to configure username/password to authenticate to the STS. If <code>policy-reference-uri</code> in the client " oracle/sts_trust_config_client_template " points to a username-based policy, then you configure the <code>sts.auth.user.csf.key</code> 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 <code>policy-reference-uri</code> in the client " oracle/sts_trust_config_client_template " points to an x509-based policy, then you configure the <code>sts.auth.x509.csf.key</code> property to specify the X509 certificate for authenticating to the STS.

Table C-7 (Cont.) Properties Used by the Predefined Assertions

Property	Description
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 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.
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 OWSM 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.
user.roles.include	SOAP roles to be included. This value defaults to false.

C.7.2 Example of orawsp:Property Element

```
<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>
```

C.8 orawsp:Description Element

The <orawsp:Description> element provides a description of the property.

C.8.1 Example of orawsp:Description Element

```
<orawsp:Description>My description.</orawsp:Description>
```

C.9 orawsp:Value Element

The <orawsp:Value> element provides a list of valid values for the property.

C.9.1 Example of orawsp:Value Element

```
<orawsp:Value>ultimateReceiver</orawsp:Value>
```

C.10 orawsp:guard Element

The <orawsp:guard> element defines the resource, action, and constraint match values.

C.10.1 Examples of orawsp:guard Element

```
<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>
```

C.11 orawsp:resource-match Element

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`.

C.11.1 Examples of orawsp:resource-match

```
<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>
```

C.12 orawsp:action-match Element

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.

C.12.1 Examples of orawsp:action-match Element

```
<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>
```

C.13 orawsp:constraint-match Element

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.

C.13.1 Example of orawsp:constraint-match Element

```
<orawsp:guard>
<orawsp:constraint-match>${!(messageContext.authenticationMethod == 'SAML_SV' ||
  messageContext.requestOrigin == 'internal')}
  </orawsp:constraint-match>
</orawsp:guard>
```

C.14 oralgp:Logging Element

The `<orasp:Logging>` element defines the logging policy.

The `<orasp:Logging>` element contains the following subelements:

- [oralgp:msg-log Element](#)
- [orawsp:bindings Element](#)

C.14.1 Example of oralgp:Logging Element

The following example shows the `oralgp:Logging` element.

```
<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>
```

```
<orasp:bindings>
  <orasp:Config orasp:name="added-from-em"/>
</orasp:bindings>
</oralgp:Logging>
```

C.15 orasp:binding-authorization Element

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:

- [orasp:bindings Element](#)
- [orasp:guard Element](#)

It also contains **one** of the following subelements:

- [orasp:denyAll Element](#)
- [orasp:permitAll Element](#)
- [orasp:role Element](#)

C.15.1 Example of orasp:binding-authorization Element

```
<orasp:binding-authorization orasp:Enforced="true" orasp:Silent="true"
orasp:category="security/authorization"
orasp:name="J2EE services Authorization">
  <orasp:denyAll/>
  <orasp:bindings>
    <orasp:Config orasp:configType="declarative" orasp:name="AuthzConfig"/>
  </orasp:bindings>
  <orasp:guard/>
</orasp:binding-authorization>
```

C.16 orasp:binding-permission-authorization Element

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 Element](#)
- [orasp:bindings Element](#)
- [orasp:guard Element](#)

C.16.1 Example of orasp:binding-permission-authorization Element

```
<orasp:binding-permission-authorization orasp:Enforced="true"
orasp:Silent="true" orasp:category="security/authorization"
orasp:name="J2EE Permission Based Authorization">
  <orasp:check-permission/>
  <orasp:bindings>
    <orasp:Config orasp:configType="declarative"
orasp:name="BindingPermissionAuthzConfig">
      <orasp:PropertySet orasp:name="perms-authz-properties">
        <orasp:Property orasp:contentType="optional" orasp:name="resource"
orasp: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>

```

C.17 orasp:coreid-security Element

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 Element](#)
- [orawsp:bindings Element](#)

C.17.1 Example of orasp:coreid-security Element

```

<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>

```

C.18 orasp:http-security Element

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 Element](#)
- [orasp:require-tls Element](#)

- [orasp:bindings Element](#)

C.18.1 Example of orasp:http-security Element

```
<orasp:http-security orasp:Enforced="true" orasp:Silent="true"
  orasp:category="security/authentication, security/msg-protection"
  orasp:name="Http over SSL Security">
  <orasp:auth-header orasp:mechanism="basic"/>
  <orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
  <orasp:bindings>
    <orasp:Config orasp:configType="declarative" orasp:name="HttpConfig">
      <orasp:PropertySet orasp:name="standard-security-properties">
        <orasp:Property orasp:contentType="constant" orasp:name="realm"
          orasp:type="string">
          <orasp:Value>owsm</orasp:Value>
        </orasp:Property>
        <orasp:Property orasp:contentType="constant" orasp:name="role"
          orasp:type="string">
          <orasp:Value>ultimateReceiver</orasp:Value>
        </orasp:Property>
      </orasp:PropertySet>
    </orasp:Config>
  </orasp:bindings>
</orasp:http-security>
```

C.19 orasp:kerberos-security Element

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 Element](#)
- [orasp:bindings Element](#)
- [orasp:msg-security Element](#)

C.19.1 Example of orasp:kerberos-security Element

```
<orasp:kerberos-security orasp:Enforced="true" orasp:Silent="false"
  orasp:category="security/authentication" orasp:name="WSS Kerberos Token">
  <orasp:kerberos-token orasp:is-encrypted="false" orasp:is-signed="false"
    orasp:type="gss-apreq-v5"/>
  <orasp:bindings>
    <orasp:Config orasp:configType="declarative"
      orasp:name="KerberosSecurityConfig"/>
  </orasp:bindings>
</orasp:kerberos-security>
```

C.20 orasp:sca-component-authorization Element

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:

- [orasp:bindings Element](#)

It also contains **one** of the following subelements:

- [orasp:denyAll Element](#)
- [orasp:permitAll Element](#)
- [orasp:role Element](#)

C.20.1 Example of orasp:sca-component-authorization Element

```
<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>
```

C.21 orasp:sca-component-permission-authorization Element

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 Element](#)
- [orawsp:bindings Element](#)
- [orawsp:guard Element](#)

C.21.1 Example of orasp:sca-component-permission-authorization Element

```
<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>

```

C.22 orasp:sts-trust-config Element

The <orasp:sts-trust-config> element provides a mechanism to invoke the STS for token exchange.

It contains the following subelements:

- [orawsp:bindings Element](#)

C.22.1 orasp:sts-trust-config Element Attributes

The following table summarizes the attributes of the <orasp:sts-trust-config> element.

Table C-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, <code>http://host:port/context-root/service1</code> .
port-endpoint	The endpoint of the STS web service. For a WSDL 2.0 STS, the format is specified as <code>target-namespace#wSDL.endpoint(service-name/port-name)</code> . For example, <code>http://samples.otn.com.LoanFlow#wSDL.endpoint(LoanFlowService/LoanFlowPort)</code> For a WSDL 1.1 STS, the format is specified as <code>targetnamespace#wSDL11.endpoint(servicename/portname)</code> . For example, <code>http://samples.otn.com.LoanFlow#wSDL11.endpoint(LoanFlowService/LoanFlowPort)</code> .
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 <code>sts-csf-key</code> .

C.22.2 Example of orasp:sts-trust-config Element

```

<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"

```



```

    orasp:Silent="true" orasp:category="security/sts-config" orasp:name="STS
    Trust Configuration">
<orasp:bindings>
<orasp:Config orasp:configType="declarative" orasp:name="StsTrustConfig">
<orasp:PropertySet orasp:name="standard-security-properties">
<orasp:Property orasp:contentType="constant" orasp:name="role" orasp:type="string">
<orasp:Value>ultimateReceiver</orasp:Value>
</orasp:Property>
</orasp:PropertySet>
</orasp:Config>
</orasp:bindings>
</orasp:sts-trust-config>

```

C.23 orasp:wss10-anonymous-with-certificates Element

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 Element](#)
- [orasp:msg-security Element](#)
- [orasp:bindings Element](#)

C.23.1 Example of orasp:wss10-anonymous-with-certificates Element

```

<orasp:wss10-anonymous-with-certificates orasp:Enforced="true"
    orasp:Silent="false" orasp:category="security/msg-protection"
    orasp: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>
</orasp:bindings>
    <orasp:Config orasp:configType="declarative"
        orasp:name="Wss10AnonWithCertsConfig">
        <orasp:PropertySet orasp:name="standard-security-properties">
            <orasp:Property orasp:contentType="constant" orasp:name="role"

```

```

        orasp:type="string">
        <orasp:Value>ultimateReceiver</orasp:Value>
    </orasp:Property>
</orasp:PropertySet>
</orasp:Config>
</orasp:bindings>
</orasp:wss10-anonymous-with-certificates>

```

C.24 orasp:wss10-mutual-auth-with-certificates Element

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 Element](#)
- [orasp:msg-security Element](#)
- [orasp:bindings Element](#)

C.24.1 Example of orasp:wss10-mutual-auth-with-certificates Element

```

<orasp:wss10-mutual-auth-with-certificates orasp:Enforced="true"
orasp:Silent="false" orasp:category="security/authentication,
security/msg-protection" orasp: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:msg-security>
</orasp:bindings>
<orasp:Config orasp:configType="declarative"
orasp:name="Wss10AnonWithCertsConfig">
  <orasp:PropertySet orasp:name="standard-security-properties">
    <orasp:Property orasp:contentType="constant" orasp:name="role"
orasp:type="string">
      <orasp:Value>ultimateReceiver</orasp:Value>
    </orasp:Property>
  </orasp:PropertySet>

```

```

    </orasp:Config>
  </orasp:bindings>
</orasp:wss10-mutual-auth-with-certificates>

```

C.25 orasp:wss10-saml-hok-with-certificates Element

The <orasp:wss10-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 Element](#)
- [orasp:x509-token Element](#)
- [orasp:msg-security Element](#)
- [orasp:bindings Element](#)

C.25.1 Example of orasp:wss10-saml-hok-with-certificates Element

```

<orasp:wss10-saml-hok-with-certificates orasp:Enforced="true"
orasp:Silent="false" orasp:category="security/authentication,
security/msg-protection" orasp: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:msg-security>
</orasp:bindings>
  <orasp:Config orasp:configType="declarative"
    orasp:name="Wss10SamlHOKWithCertsConfig">
    <orasp:PropertySet orasp:name="standard-security-properties">
      <orasp:Property orasp:name="keystore.recipient.alias"
        orasp:type="string">
        <orasp:Value>orakey</orasp:Value>
      </orasp:Property>
    </orasp:PropertySet>
  </orasp:Config>

```

```

<orasp:Property orasp:contentType="optional"
  orasp:name="saml.issuer.name" orasp:type="string">
  <orasp:Value>www.oracle.com</orasp:Value>
</orasp:Property>
<orasp:Property orasp:contentType="optional"
  orasp:name="user.roles.include" orasp:type="string">
  <orasp:Value>>false</orasp:Value>
</orasp:Property>
<orasp:Property orasp:contentType="optional"
  orasp:name="saml.assertion.filename" orasp:type="string">
  <orasp:Value>temp</orasp:Value>
</orasp:Property>
</orasp:PropertySet>
</orasp:Config>
</orasp:bindings>
</orasp:wss10-saml-hok-with-certificates>

```

C.26 orasp:wss10-saml-token Element

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 Element](#)
- [orasp:bindings Element](#)

C.26.1 Example of orasp:wss10-saml-token Element

```

<orasp:wss10-saml-token orasp:Enforced="true" orasp:Silent="false"
  orasp:category="security/authentication" orasp:name="WSecurity SAML Token">
  <orasp:saml-token orasp:confirmation-type="sender-vouches"
    orasp:is-encrypted="false" orasp:is-signed="false" orasp:version="1.1"/>
  <orasp:bindings>
    <orasp:Config orasp:configType="declarative"
      orasp:name="WssSamlTokenConfig">
      <orasp:PropertySet orasp:name="standard-security-properties">
        <orasp:Property orasp:contentType="constant" orasp:name="role"
          orasp:type="string">
          <orasp:Value>ultimateReceiver</orasp:Value>
        </orasp:Property>
      </orasp:PropertySet>
    </orasp:Config>
  </orasp:bindings>
</orasp:wss10-saml-token>

```

C.27 orasp:wss10-saml-with-certificates Element

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 Element](#)
- [orasp:x509-token Element](#)
- [orasp:msg-security Element](#)

- [orasp:bindings Element](#)

C.27.1 Example of orasp:wss10-saml-with-certificates Element

```

<orasp:wss10-saml-with-certificates orasp:Enforced="true"
  orasp:Silent="false" orasp:category="security/authentication,
  security/msg-protection" orasp: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>
</orasp:wss10-saml-with-certificates>
<orasp:bindings>
  <orasp:Config orasp:configType="declarative"
    orasp:name="Wss10SamlWithCertsConfig">
    <orasp:PropertySet orasp:name="standard-security-properties">
      <orasp:Property orasp:contentType="constant" orasp:name="role"
        orasp:type="string">
        <orasp:Value>ultimateReceiver</orasp:Value>
      </orasp:Property>
    </orasp:PropertySet>
  </orasp:Config>
</orasp:bindings>

```

C.28 orasp:wss10-username-with-certificates Element

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 Element](#)
- [orasp:x509-token Element](#)
- [orasp:msg-security Element](#)
- [orasp:bindings Element](#)

C.28.1 Example of orasp:wss10-username-with-certificates Element

```

<orasp:wss10-username-with-certificates orawsp:Enforced="true"
  orasp: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>
</orasp:wss10-username-with-certificates>
  <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>

```

C.29 orasp:wss11-anonymous-with-certificates Element

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 Element](#)
- [orasp:msg-security Element](#)
- [orawsp:bindings Element](#)

C.29.1 Example of orasp:wss11-anonymous-with-certificates Element

```

<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>

```

C.30 orasp:wss11-mutual-auth-with-certificates Element

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 Element](#)
- [orasp:msg-security Element](#)
- [orawsp:bindings Element](#)

C.30.1 Example of orasp:wss11-mutual-auth-with-certificates Element

```

<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>

```

C.31 orasp:wss11-saml-with-certificates Element

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 Element](#)
- [orasp:x509-token Element](#)
- [orasp:msg-security Element](#)
- [orawsp:bindings Element](#)

C.31.1 Example of orasp:wss11-saml-with-certificates Element

```

<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>

```

C.32 orasp:wss11-sts-issued-token-with-certificates Element

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 Element](#)
- [orasp:x509-token Element](#)
- [orasp:msg-security Element](#)
- [orawsp:bindings Element](#)

C.32.1 orasp:wss11-sts-issued-token-with-certificates Element Attributes

The following table summarizes the attributes of the <orasp:wss11-sts-issued-token-with-certificates> element.

Table C-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 OWSM 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.

C.32.2 Example of orasp:wss11-sts-issued-token-with-certificates Element

```
<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>
<orasp:bindings>
<orasp:Config orasp:configType="declarative"
  orasp:name="Wss11StsIssuedTokenWithCertsConfig">
<orasp:PropertySet orasp:name="standard-security-properties">
<orasp:Property orasp:contentType="optional"
  orasp:name="sts.auth.user.csf.key" orasp:type="string">
<orasp:Value/>
</orasp:Property>
<orasp:Property orasp:contentType="optional"
  orasp:name="sts.auth.x509.csf.key" orasp:type="string">
<orasp:Value>enc-csf-key</orasp:Value>
</orasp:Property>
<orasp:Property orasp:name="on.behalf.of" orasp:type="boolean">
<orasp:Value>>false</orasp:Value>
</orasp:Property>
<orasp:Property orasp:contentType="optional"
  orasp:name="sts.auth.on.behalf.of.csf.key" orasp:type="string">
<orasp:Value/>
</orasp:Property>
<orasp:Property orasp:name="keystore.recipient.alias" orasp:type="string">
<orasp:Value>orakey</orasp:Value>
</orasp:Property>
<orasp:Property orasp:contentType="optional" orasp:name="keystore.enc.csf.key"
  orasp:type="string">
<orasp:Value/>
</orasp:Property>
<orasp:Property orasp:contentType="optional"
  orasp:name="sts.auth.service.principal.name" orasp:type="string">
<orasp:Value>HOST/localhost@EXAMPLE.COM</orasp:Value>
</orasp:Property>
<orasp:Property orasp:contentType="optional"
  orasp:name="sts.auth.keytab.location" orasp:type="string">
<orasp:Value/>
</orasp:Property>
<orasp:Property orasp:contentType="optional"
  orasp:name="sts.auth.caller.principal.name" orasp:type="string">
<orasp:Value/>
</orasp:Property>
</orasp:PropertySet>
</orasp:Config>
</orasp:bindings>
</orasp:wss11-sts-issued-token-with-certificates>

```

C.33 orasp:wss11-username-with-certificates Element

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 Element](#)
- [orasp:x509-token Element](#)
- [orasp:msg-security Element](#)
- [orawsp:bindings Element](#)

C.33.1 Example of orasp:wss11-username-with-certificates Element

```
<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>
```

C.34 orasp:wss-saml-token-bearer-over-ssl Element

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 Element](#)
- [orasp:require-tls Element](#)
- [orawsp:bindings Element](#)

C.34.1 Example of orasp:wss-saml-token-bearer-over-ssl Element

```
<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>
```

C.35 orasp:wss-saml-token-over-ssl Element

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 Element](#)
- [orasp:require-tls Element](#)
- [orawsp:bindings Element](#)

C.35.1 Example of orasp:wss-saml-token-over-ssl Element

```
<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"

```

```

    orasp:name="saml.issuer.name" orasp:type="string">
      <orasp:Value>www.oracle.com</orasp:Value>
    </orasp:Property>
    <orasp:Property orasp:contentType="optional"
      orasp:name="user.roles.include" orasp:type="string">
      <orasp:Value>>false</orasp:Value>
    </orasp:Property>
  </orasp:PropertySet>
</orasp:Config>
</orasp:bindings>
</orasp:wss-saml-token-over-ssl>

```

C.36 orasp:wss-sts-issued-token-over-ssl Element

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 Element](#)
- [orasp:require-tls Element](#)
- [orasp:bindings Element](#)

C.36.1 orasp:wss-sts-issued-token-over-ssl Element Attributes

The following table summarizes the attributes of the <orasp:wss-sts-issued-token-over-ssl> element.

Table C-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 OWSM 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.

C.36.2 Example of orasp:wss-sts-issued-token-over-ssl Element

```

<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"

```

```

orasp:category="security/authentication, security/msg-protection"
orasp: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"/>
<orasp:bindings>
<orasp:Config orasp:configType="declarative"
orasp:name="WssStsIssuedTokenOverSSLConfig">
<orasp:PropertySet orasp:name="standard-security-properties">
<orasp:Property orasp:contentType="constant" orasp:name="role"
orasp:type="string">
<orasp:Value>ultimateReceiver</orasp:Value>
</orasp:Property>
</orasp:PropertySet>
</orasp:Config>
</orasp:bindings>
</orasp:wss-sts-issued-token-over-ssl>

```

C.37 orasp:wss-username-token Element

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 Element](#)
- [orasp:bindings Element](#)

C.37.1 Example of orasp:wss-username-token Element

```

<orasp:wss-username-token orasp:Enforced="true" orasp:Silent="false"
orasp:category="security/authentication"
orasp: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"/>
<orasp:bindings>
<orasp:Config orasp:configType="declarative"
orasp:name="WssUsernameTokenConfig">
<orasp:PropertySet orasp:name="standard-security-properties">
<orasp:Property orasp:contentType="constant" orasp:name="role"
orasp:type="string">
<orasp:Value>ultimateReceiver</orasp:Value>
</orasp:Property>
</orasp:PropertySet>
</orasp:Config>
</orasp:bindings>
</orasp:wss-username-token>

```

C.38 orasp:wss-username-token-over-ssl Element

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 Element](#)
- [orasp:require-tls Element](#)
- [orawsp:bindings Element](#)

C.38.1 Example of orasp:wss-username-token-over-ssl Element

```
<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>
```

C.39 rm:RMAssertion Element

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 Element](#)

C.39.1 Example of rm:RMAssertion Element

```
<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.PolicyDescriptionBundle_0
  racle/wsm10_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/MessageStore</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>

```

C.40 wsaw:UsingAddressing Element

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 Element](#)

C.40.1 Example of wsaw:UsingAddressing Element

```

<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>

```

C.41 wsoma:OptimizedMimeSerialization Element

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 Element](#)

C.41.1 Example of wsoma:OptimizedMimeSerialization Element

```
<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>
```

C.42 oralgp:fault Element

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.

C.42.1 Example of oralgp:fault Element

The following is an example of oralgp:fault element.

```
<oralgp:msg-log>
  <oralgp:request>all</oralgp:request>
  <oralgp:response>all</oralgp:response>
  <oralgp:fault>all</oralgp:fault>
</oralgp:msg-log>
```

C.43 oralgp:request Element

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.

C.43.1 Example of oralgp:request Element

```
<oralgp:msg-log>
  <oralgp:request>all</oralgp:request>
  <oralgp:response>all</oralgp:response>
  <oralgp:fault>all</oralgp:fault>
</oralgp:msg-log>
```

C.44 oralgp:response Element

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.

C.44.1 Example of oralgp:response Element

```
<oralgp:msg-log>
  <oralgp:request>all</oralgp:request>
  <oralgp:response>all</oralgp:response>
  <oralgp:fault>all</oralgp:fault>
</oralgp:msg-log>
```

C.45 oralgp:msg-log Element

The <oralgp:msg-log> element configures logging for the request, response, and fault messages. The <oralgp:msg-log> element contains the following subelements:

- [orasp:request Element](#)
- [oralgp:response Element](#)
- [oralgp:fault Element](#)

C.45.1 Example of oralgp:msg-log Element

```
<oralgp:msg-log>
  <oralgp:request>all</oralgp:request>
  <oralgp:response>all</oralgp:response>
  <oralgp:fault>all</oralgp:fault>
</oralgp:msg-log>
```

C.46 orasp:attachment Element

The <orasp:attachment> element defines the attachment information.

C.46.1 orasp:attachment Element Attributes

The following table summarizes the attributes of the <orasp:attachment> element.

Table C-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.

C.46.2 Example of orasp:attachment Element

```
<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>
```

C.47 orasp:auth-header Element

The <orasp:auth-header> element specifies the name of the authentication header.

C.47.1 orasp:auth-header Element Attributes

The following table summarizes the attribute of the <orasp:auth-header> element.

Table C-12 Attributes of <orasp:auth-header> Element

Attribute	Description
mechanism	Authentication mechanism. Valid values include: <ul style="list-style-type: none"> • basic—Client authenticates itself by transmitting the username and password. • digest—Not supported in this release. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. • cert—Client authenticates itself by transmitting a certificate. • custom—Custom authentication mechanism. • spnego—Client authentication mechanism determined by SPNEGO negotiaion.

C.47.2 Example of orasp:auth-header Element

```
<orasp:auth-header orasp:mechanism="basic"/>
```

C.48 orasp:body Element

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/>.

C.48.1 Example of orasp:body Element

```
<orasp:request>
  <orasp:signed-parts>
```

```

    <orasp:body/>
  </orasp:signed-parts>
  <orasp:encrypted-parts>
    <orasp:body/>
  </orasp:encrypted-parts>
</orasp:request>

```

C.49 orasp:check-permission Element

The <orasp:check-permission> element specifies that permissions are to be checked.

C.49.1 Example of orasp:check-permission Element

```

<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>

```

C.50 orasp:coreid-token Element

The <orasp:coreid-token> element defines the OAM token.

C.50.1 orasp:coreid-token Element Attributes

The following table summarizes the attributes of the <orasp:coreid-token> element.

Table C-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.

C.50.2 Example of orasp:coreid-token Element

```

<orasp:coreid-token orasp:is-encrypted="false" orasp:is-signed="false"/>

```

C.51 orasp:denyAll Element

The <orasp:denyAll> element denies all users with any roles.

C.51.1 Example of orasp:denyAll Element

```

<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>
```

C.52 orasp:element Element

The <orasp:element> element defines a header or body element that is signed or encrypted.

C.52.1 orasp:element Element Attributes

The following table summarizes the attributes of the <orasp:element> element.

Table C-14 Attributes of <orasp:element> Element

Attribute	Description
name	Name of the header or body element.
namespace	Namespace.

C.52.2 Example of orasp:element Element

```
<orasp:signed-elements>
  <orasp:element orasp:name="BodyElement"
    orasp:namespace="http://www.w3.org/2005/08/addressing">n/a</orasp:element>
</orasp:signed-elements>
```

C.53 orasp:encrypted-elements Element

The <orasp: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 <orasp:encrypted-parts> element contains the following subelement:

- [orasp:element Element](#)

C.53.1 Example of orasp:encrypted-elements Element

```
<orasp:encrypted-elements>
  <orasp:element orasp:name="Myhead"
    orasp:namespace="http://www.w3.org/2005/08/addressing">n/a</orasp:element>
</orasp:encrypted-elements>
```

C.54 orasp:encrypted-parts Element

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 Element](#)
- [orasp:header Element](#)
- [orasp:attachment Element](#)

C.54.1 Example of orasp:encrypted-parts Element

```
<orasp:request>
  <orasp:signed-parts>
    <orasp:body/>
  </orasp:signed-parts>
  <orasp:encrypted-parts>
    <orasp:body/>
  </orasp:encrypted-parts>
</orasp:request>
```

C.55 orasp:fault Element

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 Element](#)
- [orasp:encrypted-parts Element](#)

C.55.1 Example of orasp:fault Element

```
<orasp:response>
  <orasp:signed-parts>
    <orasp:body/>
  </orasp:signed-parts>
  <orasp:encrypted-parts>
    <orasp:body/>
  </orasp:encrypted-parts>
</orasp:response>
```

C.56 orasp:header Element

The <orasp:header> element defines a header element.

C.56.1 orasp:header Element Attributes

The following table summarizes the attributes of the <orasp:header> element.

Table C-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 .

C.56.2 Example of orasp:header Element

```
<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>
```

C.57 orasp:issued-token Element

The <orasp:issued-token> element enforces token characteristics.

C.57.1 orasp:issued-token Element Attributes

The following table summarizes the attributes of the <orasp:issued-token> element.

Table C-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.

C.57.2 Example of orasp:issued-token Element

```
<orasp:issued-token orasp:require-external-reference="true"
orasp:require-internal-reference="true" orasp:use-derived-keys="false">
```

C.58 orasp:kerberos-token Element

The <orasp:kerberos-token> element defines the kerberos token.

C.58.1 orasp:kerberos-token Element Attributes

The following table summarizes the attributes of the <orasp:kerberos-token> element.

Table C-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).
use-derived-keys	Flag that specifies whether derived keys are required. Valid values are true or false.

C.58.2 Example of orasp:kerberos-token Element

```
<orasp:kerberos-token orasp:is-encrypted="false" orasp:is-signed="false"
orasp:type="gss-apreq-v5" orasp:use-derived-keys="false"/>
```


C.59 orasp:msg-security Element

The <orasp: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 Element](#)
- [orasp:response Element](#)
- [orasp:fault Element](#)

C.59.1 orasp:msg-security Element Attributes

The following table summarizes the attributes of the <orasp:msg-security> element.

Table C-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 ".
confirm-signature	Flag that specifies whether to send a signature confirmation back to the client. Valid values include true or false.
encrypt-signature	Flag that specifies whether to send an encryption confirmation back to the client. Valid values include 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-encrypt	Flag that specifies whether to sign the message before encrypting the message.
use-derived-keys	Flag that specifies whether to use derived keys.

C.59.2 Example of orasp:msg-security Element

```
<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:msg-security>
```

```
</orasp:response>  
<orasp:fault/>  
</orasp:msg-security>
```

C.60 orasp:permitAll Element

The <orasp:permitAll> element permits all users with any roles.

C.60.1 Example of orasp:permitAll Element

```
<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>
```

C.61 orasp:request Element

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 Element](#)
- [orasp:encrypted-parts Element](#)

C.61.1 Example of orasp:request Element

```
<orasp:request>  
  <orasp:signed-parts>  
    <orasp:body/>  
  </orasp:signed-parts>  
  <orasp:encrypted-parts>  
    <orasp:body/>  
  </orasp:encrypted-parts>  
</orasp:request>
```

C.62 orasp:require-tls Element

The <orasp:require-tls> element specifies whether two-way authentication is required.

C.62.1 orasp:require-tls Element Attributes

The following table summarizes the attributes of the <orasp:require-tls> element.

Table C-19 Attributes of <orasp: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 flag is not valid for RESTful web service and client endpoints.
mutual-auth	Flag that specifies whether two-way authentication is required. Valid values include true or false.

C.62.2 Example of orasp:require-tls Element

```
<orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
```

C.63 orasp:response Element

The <orasp:response> element defines the message body elements that are signed and encrypted in the response message. The <orasp:response> element contains the following subelements:

- [orasp:signed-parts Element](#)
- [orasp:encrypted-parts Element](#)

C.63.1 Example of orasp:response Element

```
<orasp:response>
  <orasp:signed-parts>
    <orasp:body/>
  </orasp:signed-parts>
  <orasp:encrypted-parts>
    <orasp:body/>
  </orasp:encrypted-parts>
</orasp:response>
```

C.64 orasp:role Element

The <orasp:role> element defines the roles that are permitted access.

C.64.1 orasp:role Element Attribute

The following table summarizes the attribute of the <orasp:role> element.

Table C-20 Attributes of <orasp:role> Element

Attribute	Description
name	Name of the role. Valid roles include: <ul style="list-style-type: none"> • Monitor • AdminChannelUsers • Administrators • OracleSystemGroup • Operators • CrossDomainConnectors • Deployers • AppTesters

C.64.2 Example of orasp:role Element

```
<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>
```

C.65 orasp:saml-token Element

The <orasp:saml-token> element configures the SAML token.

C.65.1 orasp:saml-token Element Attributes

The following table summarizes the attributes of the <orasp:saml-token> element.

Table C-21 Attributes of <orasp:saml-token> Element

Attribute	Description
confirmation-type	Confirmation type. Valid values include: sender-vouches and holder-of-key. <ul style="list-style-type: none"> • 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.

C.65.2 Example of orasp:saml-token Element

```
<orasp:saml-token orasp:confirmation-type="holder-of-key"
  orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
```

C.66 orasp:signed-elements Element

The <orasp: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 <orasp:signed-elements> element contains the following subelement:

- [orasp:element Element](#)

C.66.1 Example of orasp:signed-elements Element

```
<orasp:signed-elements>
  <orasp:element orasp:name="Myhead"
    orasp:namespace="http://www.w3.org/2005/08/addressing">n/a</orasp:element>
</orasp:signed-elements>
```

C.67 orasp:signed-parts Element

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 Element](#)
- [orasp:header Element](#)
- [orasp:attachment Element](#)

C.67.1 Example of orasp:signed-parts Element

```
<orasp:request>
  <orasp:signed-parts>
    <orasp:body/>
  </orasp:signed-parts>
  <orasp:encrypted-parts>
    <orasp:body/>
  </orasp:encrypted-parts>
</orasp:request>
```

C.68 orasp:username-token Element

The <orasp:username-token> element configures the SAML token.

C.68.1 orasp:username-token Element Attributes

The following table summarizes the attributes of the <orasp:username-token> element.

Table C-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: <ul style="list-style-type: none"> • none—No password. • plaintext—Unencrypted password in clear text. • digest— Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest.

C.68.2 Example of orasp:username-token Element

```
<orasp:username-token
  orasp:add-created="false"
  orasp:add-nonce="false"
  orasp:is-encrypted="true"
  orasp:is-signed="true"
  orasp:password-type="plaintext"/>
```

C.69 orasp:x509-token Element

The <orasp:x509-token> element defines the x.509 digital certificate.

C.69.1 orasp:x509-token Element Attributes

The following table summarizes the attributes of the <orasp:x509-token> element.

Table C-23 Attributes of <orasp:x509-token> Element

Attribute	Description
sign-key-ref-mech	Mechanism used when signing the request. Valid values include: <ul style="list-style-type: none"> • 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.
use-pkipath	Flag that specifies whether X509PKIPathV1 tokens should be processed and propagated. Valid values include true or false.

C.69.2 Example of orasp:x509-token Element

```
<orasp:x509-token orasp:enc-key-ref-mech="thumbprint"
  orasp:is-encrypted="false" orasp:is-signed="true"
  orasp:sign-key-ref-mech="direct" orasp:use-pkipath="false"/>
```

C.70 orawsp:Description Element

The <orawsp:Description> element provides a description of the property.

C.70.1 Example of orawsp:Description Element

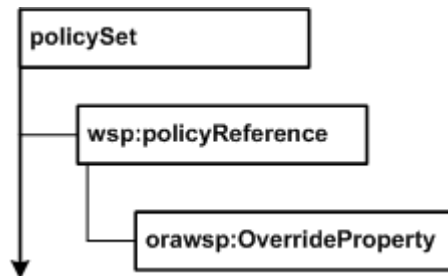
```
<orawsp:Description>Valid IP Values</orawsp:Description>
```

D

Schema Reference for Web Services Policy Sets

This appendix provides the XML schema for reference when creating a policy set file. The following graphic describes the element hierarchy of the policy set document.

Figure D-1 Element Hierarchy of the Policy Set



This appendix includes the following topics:

- [policySet Element](#)
- [wsp:policyReference Element](#)
- [orawsp:OverrideProperty Element](#)

D.1 policySet Element

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 [Figure D-1](#).

policySet Element Attributes

The following section summarizes the policy set attributes, including the Oracle extensions.

Table D-1 Attributes of Policy Set Element

Attribute	Description
name	Name of the policy set.
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.

Table D-1 (Cont.) Attributes of Policy Set Element

Attribute	Description
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.

D.2 wsp:policyReference Element

Use the `wsp:policyReference` element to associate a policy set with one or more policies.

This element contains the [orawsp:OverrideProperty Element](#).

wsp:policyReference Element Attributes

The following table summarizes the attributes of the `<wsp:policyReference>` element.

Table D-2 Attributes of `<wsp:policyReference>` Element

Attribute	Description
URI	OWSM policy URI to be associated with the policy set.
category	Category of the policy. Valid values include: security, mtom, wsrn, addressing, and management.
status	Status of the policy reference. Valid values include: enabled and disabled.

policySet Element Example

The following example shows 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".

```
<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>
```

D.3 orawsp:OverrideProperty Element

Use the `<orawsp:OverrideProperty>` element to specify a configuration override associated with a policy attachment in a policy set.

orawsp:OverrideProperty Element Example

```
<orawsp:OverrideProperty name="csf-key" value="orakey" />
```

E

Oracle Web Services Manager Introspection Plug-in for Oracle Virtual Assembly Builder

This appendix describes the Oracle Web Services Manager (OWSM) introspection plug-in for Oracle Virtual Assembly Builder.

It includes the following topics:

- [About the OWSM Introspection Plug-in for Oracle Virtual Assembly Builder](#)
- [Understanding the OWSM Introspection Plug-in](#)

E.1 About the OWSM Introspection Plug-in for Oracle Virtual Assembly Builder

Oracle Virtual Assembly Builder is a tool for virtualizing installed Oracle components, modifying those components, and then deploying them into an Oracle VM environment.

Using Oracle Virtual Assembly Builder, you can capture the configuration of existing software components in artifacts called software *appliances*. Appliances can then be grouped, and their relationships defined into artifacts called software *assemblies*, which provide a blueprint describing a complete multi-tier application topology.

The OWSM introspection extension for Oracle Virtual Assembly Builder extends the functionality of the Oracle WebLogic Server Introspector, as described in "Using the Plug-in for Oracle Virtual Assembly Builder" in *Administering Server Environments for Oracle WebLogic Server*. The plug-in examines the configuration of OWSM-specific artifacts configured as part of a WebLogic domain.

The OWSM introspection plug-in extension works with Oracle WebLogic Server 14c version 14.1.2.0.

E.2 Understanding the OWSM Introspection Plug-in

Learn how to use the OWSM introspection plug-in.

It includes the following topics:

- [OWSM Introspection Plug-in Parameter](#)
- [OWSM Introspection Plug-in Reference System Prerequisites](#)
- [OWSM Introspection Plug-in Usage Requirements](#)
- [OWSM Introspection Plug-in Resulting Artifact Type](#)
- [OWSM Introspection Plug-in Wiring](#)
- [OWSM Introspection Plug-in Wiring Properties](#)
- [OWSM Introspection Plug-in Appliance Properties](#)
- [OWSM Introspection Plug-in Supported Template Types](#)

E.2.1 OWSM Introspection Plug-in Parameter

Oracle Virtual Assembly Builder uses the OWSM introspection plug-in parameter to check for updates before introspection.

[Table E-1](#) lists the OWSM introspection plug-in parameter. For more information about the parameters required by WebLogic Server, see "Introspection Plug-in Parameters" in *Administering Server Environments for Oracle WebLogic Server*.

Table E-1 OWSM Introspection Plug-in Parameter

Parameter	Description
oracleCommonHome	Location of the Oracle Common Home directory. This parameter is optional. If specified, Oracle Virtual Assembly Builder checks for updates to the OWSM plug-in before introspection. If not specified, the Oracle Virtual Assembly Builder does not check for updates before introspection.

E.2.2 OWSM Introspection Plug-in Reference System Prerequisites

There are no additional prerequisites beyond those defined by Oracle WebLogic Server.

For the prerequisites required by Oracle WebLogic Server, see "Reference System Prerequisites" in *Administering Server Environments for Oracle WebLogic Server*.

E.2.3 OWSM Introspection Plug-in Usage Requirements

There are no additional usage requirements for OWSM beyond those defined by Oracle WebLogic Server.

For the Oracle WebLogic Server requirements, see "Plug-in Usage Requirements" in *Administering Server Environments for Oracle WebLogic Server*. [Table E-2](#) lists and describes the supported topologies for using the OWSM introspection plug-in.



Note:

Required security artifacts with valid data, including keystores and credential stores, must be present or manually created as a post-rehydration step.

Table E-2 OWSM Introspection Plug-in Supported Topologies

Topology	Description
Single domain topology	Introspection plug-in captures artifacts stored in document repository (MDS) and domain configuration directory (<code>wsm-config.xml</code>). If the artifacts are modified in the source environment, their corresponding user properties will be added to the assembly generated by the introspection plug-in.
Multiple domain topology	Multiple domains can store data in the same policy repository (MDS). The OWSM Policy Manager application is installed on each domain and they share the same MDS. To support the multiple domain topology, MDS data for all domains is moved when any one of the domains is introspected and rehydrated.

Table E-2 (Cont.) OWSM Introspection Plug-in Supported Topologies

Topology	Description
Heterogeneous service security topology	OWSM Policy Manager application is used across heterogeneous domains. The OWSM Policy Manager application is deployed in the WebLogic Server domain and the other domains connect to it. When the domain where the OWSM Policy Manager application is installed is introspected, the artifacts stored in MDS for all domains and bootstrap configurations are captured. For other heterogeneous domains, you can manually update bootstrap configurations.

E.2.4 OWSM Introspection Plug-in Resulting Artifact Type

The OWSM plug-in does not create any resulting artifact or assembly.

An assembly is created by the Oracle WebLogic Server plug-in and the OWSM plug-in only adds properties to the root assembly. For more information, see "Resulting Artifact Type" in *Administering Server Environments for Oracle WebLogic Server*.

E.2.5 OWSM Introspection Plug-in Wiring

No additional wiring is supported beyond that provided by the Oracle WebLogic Server plug-in.

For more information, see "Wiring" in *Administering Server Environments for Oracle WebLogic Server*.

E.2.6 OWSM Introspection Plug-in Wiring Properties

No additional wiring properties are supported beyond those provided by the Oracle WebLogic Server plug-in.

E.2.7 OWSM Introspection Plug-in Appliance Properties

Know more about the system and user properties for the OWSM appliance.

The following tables describe the properties for the OWSM appliance.

 **Note:**

If a user property is not modified for the introspected domain, it will not be added to the assembly.

Table E-3 describes OWSM system properties.

 **Note:**

System properties are not editable.

Table E-3 OWSM System Properties

Name	Type	Req'd	Default	Description
oracle.common.home	String	false	none	Oracle Common Home location at time of reconfiguration.

Table E-4 describes OWSM user properties for the `sts-trust-config` policy assertions, including:

- [oracle/sts_trust_config_client_template](#)
- [oracle/sts_trust_config_service_template](#)

Table E-4 OWSM Appliance User Properties - STS Trust

Name	Type	Req'd	Default	Display Name in Assembly Builder	Description
port-uri	String	false	none	<i>policyname.port-uri</i>	Service port URL.
wSDL-uri	String	false	none	<i>policyname.wSDL-uri</i>	Service WSDL URL.

Table E-5 describes OWSM user properties for the `kerberos-security`, `wss11-kerberos-over-ssl-security`, or `spnego-http-security` policy assertions, including:

- [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](#)
- [oracle/http_spnego_token_client_template](#)
- [oracle/wss11_kerberos_token_over_ssl_client_template](#)

Table E-5 OWSM User Properties - Kerberos and SPNEGO

Name	Type	Req'd	Default	Display Name in Assembly Builder	Description
caller.principal.name	String	false	none	<i>policyname.caller.principal.name</i>	See " caller.principal.name ".
keytab.location	String	false	none	<i>policyname.keytab.location</i>	See " keytab.location ".
service.principal.name	String	false	none	<i>policyname.service.principal.name</i>	See " service.principal.name ".

Table E-6 describes OWSM appliance user properties for the `wss11-sts-issued-token-with-certificates` policy assertions, including:

- [oracle/wss11_sts_issued_saml_hok_with_message_protection_client_template](#)

- [oracle/wss11_sts_issued_saml_hok_with_message_protection_service_template](#)

Table E-6 OWSM User Properties - wss11-sts-issued-token-with-certificates

Name	Type	Req'd	Default	Display Name in Assembly Builder	Description
sts.auth.caller.principal.name	String	false	none	<i>polycyname.sts.auth.caller.principal.name</i>	See " sts.auth.caller.principal.name ".
sts.auth.keytab.location	String	false	none	<i>polycyname.sts.auth.keytab.location</i>	See " sts.auth.keytab.location ".
sts.auth.service.principal.name	String	false	none	<i>polycyname.sts.auth.service.principal.name</i>	See " sts.auth.service.principal.name ".

In OWSM, you can create configuration documents for a domain to override the configuration for that domain. These documents contain properties which might change on the target environment. For more information, see "[Managing Oracle Web Services Manager Domain Configuration](#)".

If you specified bootstrap properties during installation of OWSM, an OWSM agent instance uses the bootstrap connection information (in the `wsm-config.xml` file in the `fmwconfig` directory) to connect to the OWSM Policy Manager. For more information, see "[Managing Oracle Web Services Manager Domain Configuration](#)".

[Table E-7](#) describes OWSM user properties for configuration bootstrapping.

Table E-7 OWSM Appliance User Properties - Configuration Bootstrapping

Name	Type	Req'd	Default	Display Name in Assembly Builder	Description
keystore.path	String	false	none	<i>bootstrap.ssl.keystore</i>	Configuration Manager keystore location. For more information, see " Managing Oracle Web Services Manager Domain Configuration ".
pm.url	String	false	none	<i>bootstrap.pm.url</i>	Configuration Manager PM URL, in the form <code>http://hostname:port</code> . or more information, see " Managing Oracle Web Services Manager Domain Configuration ".
truststore.path	String	false	none	<i>bootstrap.ssl.truststore</i>	Configuration Manager trust store location. For more information, see " Managing Oracle Web Services Manager Domain Configuration ".

E.2.8 OWSM Introspection Plug-in Supported Template Types

The supported template type for OWSM introspection plug-in is Oracle Enterprise Linux (OEL).

F

Twitter REST APIs

Twitter REST APIs are supported by OWSM.

The REST APIs provide programmatic access to read and write Twitter data, author a new Tweet, read author profile and follower data etc. The REST API identifies Twitter applications and users using [OAuth](#), responses are available in JSON. Two APIs (one POST and one GET) which are tested are listed below:

- Twitter REST Client sample code for POST request
- Twitter REST Client sample code for GET request

F.1 Twitter REST Client sample code for POST request

REST API identify Twitter applications and users, using OAuth. The responses are available in JSON. The following is a REST Client sample code for POST request.

```
package sample.rest.client;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Date;

import java.util.Set;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.ws.rs.client.Client;

import javax.ws.rs.client.ClientBuilder;

import javax.ws.rs.client.Entity;

import javax.ws.rs.client.Invocation.Builder;

import javax.ws.rs.client.WebTarget;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import javax.ws.rs.core.MultivaluedMap;
```



```
import javax.ws.rs.core.MultivaluedHashMap;

import org.glassfish.jersey.client.ClientConfig;
import oracle.wsm.metadata.feature.PolicyReferenceFeature;
import oracle.wsm.metadata.feature.AbstractPolicyFeature;
import oracle.wsm.metadata.feature.PolicySetFeature;
import oracle.wsm.metadata.feature.PropertyFeature;
import oracle.wsm.security.util.SecurityConstants;

public class TwitterTestClientPost extends HttpServlet {

    @Override

    protected void doGet(HttpServletRequest httpRequest, HttpServletResponse
httpResponse)

    throws ServletException, IOException {

        httpResponse.setContentType("text/plain;charset=UTF-8");

        PrintWriter out = httpResponse.getWriter();

        try {

            System.setProperty("https.proxyHost", "www-proxy.us.example.com");

            System.setProperty("https.proxyPort", "80");

            String BASE_URI = "https://api.twitter.com/1.1/statuses/update.json";

            PropertyFeature clientCsfKey = new
PropertyFeature(SecurityConstants.ConfigOverride.CO_OAUTH1_CLIENT_CSF_KEY,
"basic.client.oauth1.credentials");

            PropertyFeature accessTokenCsfKey = new
PropertyFeature(SecurityConstants.ConfigOverride.CO_OAUTH1_ACCESS_TOKEN_CSF_KEY,
"basic.token.oauth1.credentials");

            PolicyReferenceFeature clientPRF = new PolicyReferenceFeature("oracle/
http_oauth1_token_client_policy", clientCsfKey, accessTokenCsfKey);

            ClientConfig cc = new ClientConfig();

            cc.property(AbstractPolicyFeature.ABSTRACT_POLICY_FEATURE, new
PolicySetFeature(clientPRF));

            Client client = ClientBuilder.newClient(cc);

            WebTarget webTarget = client.target(BASE_URI);
```

```
        Builder request = webTarget.request(MediaType.APPLICATION_JSON_TYPE);

        MultivaluedMap<String, String> formData = new MultivaluedHashMap<String,
String>();

        String tweet = "Testing POST_Status at " + new Date();

        formData.add("status", tweet);

        Response response = request.post(Entity.form(formData));

        String textEntity = response.readEntity(String.class);

        out.println("Response status :" + response.getStatus());

        out.println("Response :" + textEntity);

                MultivaluedMap<String, Object> map = response.getHeaders();

                Set<String> headerkeys = map.keySet();

        for (String key : headerKeys) {

                out.println("Header Name :" + key);

                out.println("Header Value :" + map.get(key));

        }

    } catch (Exception e) {

        out.println("exception message :" + e);

    }

}
```

F.2 Twitter REST Client sample code for GET request

REST API identify Twitter applications and users using OAuth. The responses are available in JSON. The following is a REST Client sample code for GET request.

```
package sample.rest.client;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Date;

import java.util.Set;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation.Builder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.MultivaluedMap;
import javax.ws.rs.core.MultivaluedHashMap;
import org.glassfish.jersey.client.ClientConfig;
import oracle.wsm.metadata.feature.PolicyReferenceFeature;
import oracle.wsm.metadata.feature.AbstractPolicyFeature;
import oracle.wsm.metadata.feature.PolicySetFeature;
import oracle.wsm.metadata.feature.PropertyFeature;
import oracle.wsm.security.util.SecurityConstants;

public class TwitterTestClientGet extends HttpServlet {

    @Override

    protected void doGet(HttpServletRequest httpRequest, HttpServletResponse
httpResponse)

        throws ServletException, IOException {

        httpResponse.setContentType("text/plain;charset=UTF-8");

        PrintWriter out = httpResponse.getWriter();

        try {

            System.setProperty("https.proxyHost", "www-proxy.us.example.com");

            System.setProperty("https.proxyPort", "80");

            String BASE_URI = "https://api.twitter.com/1.1/followers/list.json?
screen_name=TwitterapiT";

            PropertyFeature clientCsfKey = new
PropertyFeature(SecurityConstants.ConfigOverride.CO_OAUTH1_CLIENT_CSF_KEY,
"basic.client.oauth1.credentials");
```

```
PropertyFeature accessTokenCsfKey = new
PropertyFeature(SecurityConstants.ConfigOverride.CO_OAUTH1_ACCESS_TOKEN_CSF_KEY,
"basic.token.oauth1.credentials");

PolicyReferenceFeature clientPRF = new PolicyReferenceFeature("oracle/
http_oauth1_token_client_policy", clientCsfKey, accessTokenCsfKey);

ClientConfig cc = new ClientConfig();

cc.property(AbstractPolicyFeature.ABSTRACT_POLICY_FEATURE, new
PolicySetFeature(clientPRF));

Client client = ClientBuilder.newClient(cc);

WebTarget webTarget = client.target(BASE_URI);

Builder request = webTarget.request(MediaType.APPLICATION_JSON_TYPE);

Response response = request.get(Response.class);

String textEntity = response.readEntity(String.class);

out.println("Response status :" + response.getStatus());

out.println("Response :" + textEntity);

MultivaluedMap <String,object> map = response.getHeaders();

Set<String> headerkeys = map.keySet();

for (String key : headerkeys) {

    out.println("Header Name :" + key);

    out.println("Header Value :" + map.get(key));

}

} catch (Exception e) {

    out.println("exception message :" + e);

}

}

}
```