Oracle® Fusion Middleware

Security and Administrator's Guide for Web Services

11*g* Release 1 (11.1.1)

B32511-02

October 2009

This document describes how to administer and secure Web services.

Oracle Fusion Middleware Security and Administrator's Guide for Web Services, 11*g* Release 1 (11.1.1)

B32511-02

Primary Author: Oracle Corporation

Contents

Preface xxv

About this Guide xxv
Audience xxv
How to Use This Guide xxv
Documentation Accessibility xxvi
Related Documents xxvii
Conventions xxvii

What's New xxix

Part I Introduction

1 Overview of Web Services Security and Administration

Web Services Security and Administration in Oracle Fusion Middleware 11*g* 1-1
Web Service Security and Administration Tasks 1-2
Securing and Administering Oracle Infrastructure Web Services 1-3
Securing and Administering WebLogic Web Services 1-4
Accessing the Security and Administration Tools 1-5
Accessing Oracle Enterprise Manager Fusion Middleware Control 1-5
Accessing Oracle WebLogic Administration Console 1-6
Accessing the Web Services Custom WLST Commands 1-6

2 Understanding Web Services Security Concepts

Securing Web Services 2-1
Transport-level Security 2-2
Application-level Security 2-2
Web Service Security Requirements 2-3
How Oracle Fusion Middleware Secures Web Services and Clients 2-3

3 Understanding Oracle WSM Policy Framework

Overview of Oracle WSM Policy Framework 3-1
What Are Policies? 3-4
Building Policies Using Policy Assertions 3-5
Attaching Policies to Subjects 3-6

How Policies are Executed 3-6
Oracle WSM Predefined Policies and Assertion Templates 3-8
Defining Multiple Policy Alternatives (OR Groups) 3-8
Overriding Security Policy Configuration 3-9
Recommended Naming Conventions for Policies 3-9

4 Examining the Rearchitecture of Oracle WSM in Oracle Fusion Middleware

How Oracle WSM 10*g* is Redesigned in Oracle Fusion Middleware 11*g* Release 1 (11.1.1) 4-1
Comparing Oracle WSM 10*g* and Oracle WSM 11*g* Policies 4-3
Comparing Oracle Application Server 10g WS-Security with Oracle WSM 11*g* 4-4
Interoperability and Upgrade 4-5

Part II Basic Administration

5 Deploying Web Services Applications

Overview 5-1
Additional Deployment Documentation Available 5-1
Deploying Web Services Applications 5-2
Undeploying a Web Services Application 5-5
Redeploying a Web Services Application 5-5

6 Administering Web Services

Viewing All Current Web Services for a Server 6-2
Using Fusion Middleware Control 6-2
Using WLST 6-2
Navigating to the Web Services Summary Page for an Application 6-4
Viewing the Web Services in Your Application 6-4
Using Fusion Middleware Control 6-4
Using WLST 6-5
Viewing the Details for a Web Service Port 6-5
Using Fusion Middleware Control 6-5
Using WLST 6-6
Viewing the Security Violations for a Web Service 6-7
Viewing Web Service Clients 6-7
Using Fusion Middleware Control 6-7
Viewing SOA References 6-7
Viewing ADF DC Web Service Clients 6-7
Viewing WebCenter Portlets 6-8
Viewing Asynchronous Web Service Callback Clients 6-8
Using WLST 6-8
Displaying the Web Service WSDL Document 6-9
Configuring the Web Service Port 6-10
Using Fusion Middleware Control 6-10
Using WLST 6-10
Configuring Asynchronous Web Services 6-12
Enabling or Disabling a Web Service 6-13

Using Fusion Middleware Control 6-13
Using WLST 6-14
Enabling or Disabling RESTful Web Services 6-14
Using Fusion Middleware Control 6-14
Using WLST 6-15
Enabling or Disabling the Display of the Web Service WSDL Document 6-15
Using Fusion Middleware Control 6-15
Using WLST 6-16
Enabling or Disabling the Exchange of Metadata 6-16
Enabling or Disabling the Web Service Test Endpoint 6-16
Using Fusion Middleware Control 6-16
Using WLST 6-17
Validating the Request Message 6-17
Setting the Size of the Request Message 6-18
Using Fusion Middleware Control 6-18
Using WLST 6-18
Enabling and Disabling MTOM 6-19
Configuring the Web Service Client 6-19
Using Fusion Middleware Control 6-20
Configuring SOA References 6-20
Configuring ADF DC Web Service Clients 6-20
Configuring Asynchronous Web Service Callback Clients 6-20
Using WLST 6-21

7 Managing Web Service Policies

Overview of Web Services Policy Management 7-1
Viewing Available Web Services Policies 7-1
Navigating to the Web Services Policies Page in Fusion Middleware Control 7-2
Displaying a List of the Available Policies Using WLST 7-2
Viewing a Web Service Policy 7-3
Searching for Web Service Policies 7-3
Creating Web Service Policies 7-4
Creating a New Web Service Policy 7-4
Creating a Web Service Policy from an Existing Policy 7-6
Importing Web Service Policies 7-7
Creating Custom Policies 7-7
Working With Assertions 7-7
Naming Conventions for Assertion Templates 7-8
Viewing an Assertion Template 7-8
Adding Assertions to a Policy 7-8
Adding an OR Group to a Policy 7-8
Configuring Assertions 7-9
Validating Web Services Policies 7-9
Validating a Policy 7-10
Editing Web Service Policies 7-10
Versioning Web Service Policies 7-11
Viewing the Version History of Web Services Policies 7-11

About the Restore and Activate Policy Options 7-12
Creating a New Version of a Web Service Policy 7-13
Restoring an Earlier Version of a Web Service Policy 7-13
Deleting Versions of a Web Service Policy 7-14
Exporting Web Service Policies 7-14
Deleting Web Service Policies 7-14
Generating Client Policies 7-15
Disabling a Policy for a Single Policy Subject 7-17
Using Fusion Middleware Control 7-17
Using WLST 7-17
Disabling a Web Service Policy for All Subjects 7-18
Analyzing Policy Usage 7-19

8 Attaching Policies to Web Services

Viewing the Policies That are Attached to a Web Service 8-1
Using Fusion Middleware Control 8-1
Using WLST 8-2
Attaching a Policy to a Single Subject 8-3
Attaching a Policy to a Web Service Using Fusion Middleware Control 8-3
Attaching a Policy to a Web Service Using WLST 8-4
Attaching a Policy to Multiple Subjects (Bulk Attachment) 8-6
Validating Policy Subjects 8-8
Attaching Policies to Web Service Clients 8-9
Using Fusion Middleware Control 8-9
Attaching Policies to SOA References 8-9
Attaching Policies to ADF DC Web Service Clients 8-9
Attaching Policies to Asynchronous Web Service Callback Clients 8-10
Using WLST 8-10
Attaching Client Policies Permitting Overrides 8-13
Clearing a Configuration Property 8-14
Attaching Web Service Policies Permitting Overrides 8-14
Configuring Server-Side Override Properties for Message Protection Policies 8-14
Setting Default Values for the Configuration Properties 8-15
Overriding the Configuration Properties When Attaching a Policy 8-16
Configuring Server-Side Override Properties for Authorization Policies 8-16
Setting Default Values for the Configuration Properties 8-16
Overriding the Configuration Properties When Attaching a Policy 8-17
Configuring User-Defined Client- or Server-Side Override Properties 8-17
Scope of User-Defined Configuration Properties 8-17
Adding a User-Defined Configuration Property 8-18
Editing a User-Defined Configuration Property 8-18
Deleting a User-Defined Configuration Property 8-19
Overriding the Configuration Properties When Attaching a Policy 8-19
Overriding Configuration Properties When Attaching a Policy Using WLST 8-20

9 Setting Up Your Environment for Policies

Configuring Keystores for SSL 9-1
Which Policies Require You to Configure SSL? 9-2
Which Policies Require You to Configure Two-Way SSL? 9-2
How to Configure a Keystore on WebLogic Server 9-3
Configuring SSL on WebLogic Server (One-Way) 9-4
Configuring SSL on WebLogic Server (Two-Way) 9-5
Configuring SSL for a Web Service Client 9-6
Configuring Two-Way SSL for a Web Service Client 9-7
Setting up the Keystore for Message Protection 9-7
Setting Up the Web Service Client Keystore at Design Time 9-8
How to Obtain a Trusted Certificate 9-9
How to Create and Use a Java Keystore 9-9
How to Create Private Keys and Load Trusted Certificates 9-9
Using Service Identity Certification Extension 9-10
Hostname Verification Included in WSDL 9-11
Enabling or Disabling Service Identity Certificate Extension and Hostname Verification 9-11
Ignoring the Service Identity Certificate Extension From the Client 9-12
Ignoring Hostname Verification from the Client 9-12
Configuring the Credential Store Provider 9-12
Configuring an Authentication Provider in WebLogic Server 9-14
What Type of WebLogic Security Authentication Providers Must You Create? 9-14
Configuring the SAML and Kerberos Login Modules 9-15
Configuring SAML 9-16
How the SAML Token is Validated 9-17
Which Authentication Provider is Used? 9-17
How to Configure SAML Web Service Client at Design Time 9-17
Configure the Username for the SAML Assertion 9-17
Including User Roles in the Assertion 9-18
How to Configure Oracle Platform Security Services (OPSS) for SAML Policies 9-18
Adding an Additional SAML Assertion Issuer Name 9-19
Using Kerberos Tokens 9-19
Configuring the KDC 9-19
Initializing and Starting the KDC 9-19
Creating Principals 9-20
Configuring the Web Service Client to Use the Correct KDC 9-20
Setting the Service Principal Name In the Web Service Client 9-21
Setting the Service Principal Name In the Web Service Client at Design Time 9-21
Configuring the Web Service to Use the Right KDC 9-21
Using the Correct Keytab File in Enterprise Manager 9-21
Extract and Export the Keytab File 9-22
Modify the krb5 Login Module to use the Keytab File 9-22
Authenticating the User Corresponding to the Service Principal 9-22
Creating a Ticket Cache for the Web Service Client 9-22
SAML Message Protection Use Case 9-23
What You Need to Know 9-23

Requirements of the wss11_saml_token_with_message_protection_service_policy Policy 9-24
How Are Messages Protected Via Symmetric Keys? 9-24
What Keys Must Be in the Keystore? 9-25
Multi-Domain Use Case (Keystore Hardening) 9-25
When to Override the SAML Issuer 9-26
Main Steps 9-26
Create a WebLogic Server User 9-26
Create a Java Keystore 9-27
Configure the Web Services Manager Keystore 9-28
Store the Password for the Decryption Key in the Credential Store 9-29
Attach the Policy to Your Web Service 9-29
Attach the Policy to Your Web Service Client 9-29

10 Configuring Policies

Determining Which Security Policies to Use 10-1
Protecting Messages 10-2
Message Protection Basics 10-3
Security SwA Attachments 10-4
Which Policies Offer Message Protection? 10-4
Authentication-Only Policies and Configuration Steps 10-5
oracle/wss_http_token_client_policy 10-5
Settings You Can Change 10-5
Properties You Can Configure 10-5
How to Set Up the Web Service Client 10-6
How to Set Up the Web Service Client at Design Time 10-6
oracle/wss_http_token_service_policy 10-6
Settings You Can Change 10-6
Properties You Can Configure 10-6
How to Set Up WebLogic Server 10-6
oracle/wss_username_token_client_policy 10-6
Settings You Can Change 10-7
Properties You Can Configure 10-7
How to Set Up the Web Service Client 10-7
How to Set Up the Web Service Client At Design Time 10-7
oracle/wss_username_token_service_policy 10-7
Settings You Can Change 10-7
Properties You Can Configure 10-7
How to Set Up WebLogic Server 10-7
oracle/wss10_saml_token_client_policy 10-8
Settings You Can Change 10-8
Properties You Can Configure 10-8
How to Set Up the Web Service Client 10-8
How to Set Up the Web Service Client at Design Time 10-8
oracle/wss10_saml_token_service_policy 10-8
Settings You Can Change 10-8
Properties You Can Configure 10-8

Configure the Login Module 10-8
How to Set Up WebLogic Server 10-9
oracle/wss11_kerberos_token_client_policy 10-9
Settings You Can Change 10-9
Properties You Can Configure 10-9
How to Set Up the Web Service Client 10-9
How to Set Up the Web Service Client at Design Time 10-9
oracle/wss11_kerberos_token_service_policy 10-9
Settings You Can Change 10-10
Properties You Can Configure 10-10
Configure the Login Module 10-10
How to Configure WebLogic Server 10-10
Message Protection-Only Policies and Configuration Steps 10-10
oracle/wss10_message_protection_client_policy 10-10
Settings You Can Change 10-10
Properties You Can Configure 10-10
How to Set Up the Web Service Client 10-11
How to Set Up the Web Service Client at Design Time 10-11
oracle/wss10_message_protection_service_policy 10-12
Settings You Can Change 10-13
Properties You Can Configure 10-13
How to Set Up Oracle Platform Security Services (OPSS) 10-13
oracle/wss11_message_protection_client_policy 10-13
Settings You Can Change 10-13
Properties You Can Configure 10-13
How to Configure the Web Service Client 10-13
How to Configure the Web Service Client at Design Time 10-14
oracle/wss11_message_protection_service_policy 10-15
Settings You Can Change 10-15
Properties You Can Configure 10-15
How to Set Up Oracle Platform Security Services (OPSS) 10-15
Message Protection and Authentication Policies and Configuration Steps 10-15
Configuring a Policy With an OR Group 10-16
oracle/wss_http_token_over_ssl_client_policy 10-16
Setting You Can Change 10-16
Properties You Can Configure 10-16
How to Set Up the Web Services Client 10-16
How to Set Up the Web Service Client at Design Time 10-17
oracle/wss_http_token_over_ssl_service_policy 10-17
Settings You Can Change 10-17
Properties You Can Configure 10-17
How to Set Up WebLogic Server 10-17
oracle/wss_saml_token_bearer_over_ssl_client_policy 10-18
Settings You Can Change 10-18
Properties You Can Configure 10-18
How to Set Up the Web Service Client 10-18
How to Set Up the Web Service Client at Design Time 10-18

oracle/wss_saml_token_bearer_over_ssl_service_policy 10-18
  Settings You Can Change 10-18
  Properties You Can Configure 10-18
  Configure the Login Module 10-19
  How to Set Up Oracle Platform Security Services (OPSS) 10-19
oracle/wss_saml_token_over_ssl_client_policy 10-19
  Settings You Can Change 10-19
  Properties You Can Configure 10-19
  How to Set Up the Web Service Client 10-19
  How to Set Up the Web Service Client at Design Time 10-19
oracle/wss_saml_token_over_ssl_service_policy 10-20
  Settings You Can Change 10-20
  Properties You Can Configure 10-20
  Configure the Login Module. 10-20
  How to Set Up WebLogic Server 10-20
oracle/wss_username_token_over_ssl_client_policy 10-20
  Settings You Can Change 10-20
  Properties You Can Configure 10-20
  How to Set Up the Web Service Client 10-21
  How to Set Up the Web Service Client at Design Time 10-21
oracle/wss_username_token_over_ssl_service_policy 10-21
  Settings You Can Change 10-21
  Properties You Can Configure 10-21
  How to Set Up WebLogic Server 10-21
oracle/wss10_saml_hok_token_with_message_protection_client_policy 10-22
  Settings You Can Change 10-22
  Properties You Can Configure 10-22
  How to Set Up the Web Service Client 10-22
  How to Set Up the Web Service Client at Design Time 10-22
oracle/wss10_saml_hok_token_with_message_protection_service_policy 10-23
  Configure the Login Module 10-23
  How to Set Up WebLogic Server 10-23
oracle/wss10_saml_token_with_message_integrity_client_policy 10-24
  Settings You Can Change 10-24
  Properties You Can Configure 10-24
  How to Set Up the Web Service Client 10-24
  How to Set Up the Web Service Client at Design Time 10-24
oracle/wss10_saml_token_with_message_integrity_service_policy 10-25
  Settings You Can Change 10-25
  Properties You Can Configure 10-25
  Configure the Login Module 10-25
  How to Set Up WebLogic Server 10-25
oracle/wss10_saml_token_with_message_protection_client_policy 10-25
  Settings You Can Change 10-26
  Properties You Can Configure 10-26
  How to Set Up the Web Service Client 10-26
  How to Set Up the Web Service Client at Design Time 10-26

oracle/wss10_saml_token_with_message_protection_service_policy 10-27
  Settings You Can Change 10-27
  Properties You Can Configure 10-27
  Configure the Login Module................................ 10-27
  How to Set Up WebLogic Server 10-27
oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy 10-27
  Settings You Can Change 10-28
  Properties You Can Configure 10-28
  How to Set Up the Web Service Client 10-28
  How to Set Up the Web Service Client at Design Time 10-28
oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy 10-29
  Settings You Can Change 10-29
  Properties You Can Configure 10-29
  Configure the Login Module................................ 10-29
  How to Set Up WebLogic Server 10-29
oracle/wss10_username_id_propagation_with_msg_protection_client_policy 10-30
  Settings You Can Change 10-30
  Properties You Can Configure 10-30
  How to Set Up the Web Service Client 10-30
  How to Set Up the Web Service Client at Design Time 10-30
oracle/wss10_username_id_propagation_with_msg_protection_service_policy 10-31
  Settings You Can Change 10-31
  Properties You Can Configure 10-31
  How to Set Up WebLogic Server 10-31
oracle/wss10_username_token_with_message_protection_client_policy......................... 10-32
  Settings You Can Change 10-32
  Properties You Can Configure 10-32
  How to Set Up the Web Service Client 10-32
  How to Set Up the Web Service Client at Design Time 10-32
oracle/wss10_username_token_with_message_protection_service_policy 10-33
  Settings You Can Change 10-33
  Properties You Can Configure 10-33
  How to Set Up WebLogic Server 10-33
oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy... 10-33
  Settings You Can Change 10-34
  Properties You Can Configure 10-34
  How to Set Up the Web Service Client 10-34
  How to Set Up the Web Service Client at Design Time 10-34
oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy 10-35
  Settings You Can Change 10-35
  Properties You Can Configure 10-35
  How to Set Up WebLogic Server 10-35
oracle/wss10_x509_token_with_message_protection_client_policy 10-35
  Settings You Can Change 10-36
  Properties You Can Configure 10-36
  How to Set Up the Web Service Client 10-36
  How to Set Up the Web Service Client at Design Time 10-36

oracle/wss10_x509_token_with_message_protection_service_policy 10-36
Settings You Can Change 10-37
Attributes You Can Configure 10-37
How to Set Up Oracle Platform Security Services (OPSS)... 10-37
oracle/wss11_kerberos_token_with_message_protection_client_policy............................ 10-37
Settings You Can Change 10-37
Properties You Can Configure 10-37
How to Set up the Web Service Client.. 10-37
How to Set Up the Web Service Client at Design Time 10-38
oracle/wss11_kerberos_token_with_message_protection_service_policy 10-38
Settings You Can Change 10-38
Properties You Can Configure.. 10-38
Configure the Login Module... 10-38
How to Set Up Oracle Platform Security Services (OPSS)... 10-38
oracle/wss11_saml_token_with_message_protection_client_policy................................... 10-39
Settings You Can Change 10-39
Properties You Can Configure 10-39
How to Set Up the Web Service Client 10-39
How to Set Up the Web Service Client at Design Time 10-40
oracle/wss11_saml_token_with_message_protection_service_policy 10-40
Settings You Can Change 10-40
Properties You Can Configure 10-40
Configure the Login Module... 10-40
How to Set Up Oracle Platform Security Services (OPSS)... 10-40
oracle/wss11_username_token_with_message_protection_client_policy.......................... 10-41
Settings You Can Change 10-41
Properties You Can Configure.. 10-41
How to Set Up the Web Service Client 10-41
How to Set Up the Web Service Client at Design Time 10-42
oracle/wss11_username_token_with_message_protection_service_policy 10-42
Settings You Can Change 10-42
Properties You Can Configure 10-42
How to Set Up Oracle Platform Security Services (OPSS)... 10-42
oracle/wss11_x509_token_with_message_protection_client_policy.................................. 10-43
Settings You Can Change 10-43
Properties You Can Configure 10-43
How to Set Up the Web Service Client 10-43
How to Set Up the Web Service Client at Design Time 10-43
oracle/wss11_x509_token_with_message_protection_service_policy 10-43
Settings You Can Change 10-44
Properties You Can Configure.. 10-44
How to Set Up Oracle Platform Security Services (OPSS)... 10-44
Authorization Policies and Configuration Steps 10-44
Determining Which Resources to Protect.. 10-45
How Authorization Permissions Are Determined.. 10-45
OPSS Resource Name Can Include Operation Name 10-47
oracle/binding_authorization_denyall_policy.. 10-47

Settings You Can Change 10-47
Properties You Can Configure 10-48
How to Set Up Oracle Platform Security Services (OPSS) 10-48
oracle/binding_authorization_permitall_policy 10-48
Settings You Can Change 10-48
Properties You Can Configure 10-49
How to Set Up Oracle Platform Security Services (OPSS) 10-49
oracle/binding_permission_authorization_policy 10-49
Settings You Can Change 10-49
Attributes You Can Configure 10-49
How to Set Up Oracle Platform Security Services (OPSS) 10-50
oracle/component_authorization_denyall_policy 10-50
Settings You Can Change 10-50
Properties You Can Configure 10-50
How to Set Up Oracle Platform Security Services (OPSS) 10-51
oracle/component_authorization_permitall_policy 10-51
Settings You Can Change 10-51
Properties You Can Configure 10-51
How to Set Up Oracle Platform Security Services (OPSS) 10-51
oracle/component_permission_authorization_policy 10-52
Settings You Can Change 10-52
Properties You Can Configure 10-52
How to Set Up Oracle Platform Security Services (OPSS) 10-52
WS-Addressing Policies and Configuration Steps 10-52
oracle/wsaddr_policy 10-53
How to Set Up the Web Service Client 10-53
How to Set Up the Web Service Client at Design Time 10-53
How to Set Up Oracle Platform Security Services (OPSS) 10-53
MTOM Attachment Policies and Configuration Steps 10-53
oracle/wsmtom_policy 10-53
How to Set Up the Web Service Client 10-53
How to Set Up the Web Service Client at Design Time 10-53
How to Set Up Oracle Platform Security Services (OPSS) 10-54
Reliable Messaging Policies and Configuration Steps 10-54
WS-RM Policy Properties 10-54
oracle/wsrm10_policy 10-55
How to Set Up the Web Service Client 10-55
How to Set Up the Web Service Client at Design Time 10-55
How to Set Up Oracle Platform Security Services (OPSS) 10-56
oracle/wsrm11_policy 10-56
How to Set Up the Web Service Client 10-56
How to Set Up the Web Service Client at Design Time 10-56
How to Set Up Oracle Platform Security Services (OPSS) 10-56
Management Policies and Configuration Steps 10-56
oracle/log_policy 10-56
Settings You Can Change 10-57
Properties You Can Configure 10-57

How to Set Up the Web Service or Client 10-57
How to Set Up Oracle Platform Security Services (OPSS) 10-57
Attaching Policy Files to Web Services and Clients 10-57
Using Client Programmatic Configuration Overrides 10-58
Configuration Override Example 10-65
Configuring Local Optimization 10-66

11 Testing Web Services

Testing Your Web Services 11-1
Editing the Input Arguments as XML Source 11-4
Enabling Authentication 11-4
Enabling Quality of Service Testing 11-5
Enabling HTTP Transport Options 11-6
Stress Testing the Web Service Operation 11-6
Disabling the Test Page for a Web Service 11-7

12 Monitoring the Performance of Web Services

Overview of Performance Monitoring 12-1
When Are Web Service Statistics Started or Reset? 12-1
Viewing Web Service Statistics from the Summary Page 12-2
Viewing Web Service Statistics for a Server Instance 12-2
Viewing Web Service-Specific Statistics 12-3
Viewing Endpoint-Specific Operations Statistics 12-3
Viewing Policy Security Violations for an Endpoint 12-4

Part III Advanced Administration

13 Advanced Administration

Registering Web Services 13-1
WSIL Basics 13-2
Registering a Web Service 13-2
Viewing and Editing a Registered Web Service 13-3
Unregistering a Web Service 13-3
Auditing Web Services 13-4
Configuring Audit Policies 13-5
Managing Audit Data Collection and Storage 13-6
Viewing Audit Reports 13-6
Managing the WSDL 13-7
Managing Policy Assertion Templates 13-7
Navigating to the Web Services Assertion Templates Page 13-7
Viewing an Assertion Template 13-8
Searching for an Assertion Template 13-8
Creating an Assertion Template 13-9
Exporting an Assertion Template 13-10
Importing an Assertion Template 13-10
Editing an Assertion Template 13-10

Deleting an Assertion Template 13-11
About the Metadata Services (MDS) Repository 13-11
Upgrading the Oracle WSM Policies in the MDS Repository 13-12
Adding Security to a Running Client 13-13
Managing Policy Accessor, Cache, and Interceptor Properties 13-13
Setting Up the Java Object Cache 13-14
Running the configure-joc.py Script 13-14
Deleting the OracleSystemUser Default User 13-15

14 Creating Custom Assertions

Overview of Custom Assertion Creation 14-1
Step 1: Create the Custom Assertion Class 14-1
Step 2: Create the Custom Policy File 14-3
Step 3: Create the policy-config.xml File 14-4
Step 4: Create the JAR File 14-5
Step 5: Update Your CLASSPATH 14-5
Step 6: Import the Custom Policy File 14-5
Step 7: Attach the Custom Policy to a Web Service or Client 14-5

15 Managing Horizontal Policy Migration

Overview of Horizontal Policy Migration 15-1
Migrating Policies 15-2
Migrating Policy Configuration 15-2
Migrating Keystores 15-3
Migrating Users and Groups 15-3
Migrating Credentials 15-3
Migrating Username and Password 15-3
Migrating Keystores and Encryption Key Passwords 15-4
Migrating Oracle Platform Security Services Application and System Policies 15-4
Migrating Oracle Platform Security Services Configuration 15-4
Migrating SSL 15-4
Migrating Kerberos Configuration 15-4
Migrating Assertion Templates 15-5

16 Diagnosing Problems

Diagnosing Problems with Oracle WSM Policy Manager 16-1
Diagnosing Problems Using Logs 16-3
Using Diagnostic Logs for Web Services 16-3
Setting the Log Level for Diagnostic Logs 16-3
Viewing Diagnostic Logs 16-4
Filtering Diagnostic Logs 16-6
Using Message Logs for Web Services 16-6
Configuring Message Logs 16-6
Viewing Message Logs 16-7
Filtering Message Logs 16-7
Reviewing Sample Logs 16-7

Sample Log: Oracle WSM Policy Manager Not Available 16-7
Sample Log: Security Keystore Not Configured 16-7
Sample Log: Certificate Not Available 16-8
Configuring a Diagnostic Logger for a Web Service 16-9

Part IV WebLogic Web Service Administration

17 Securing and Administering WebLogic Web Services

Steps to Secure and Administer WebLogic Web Services 17-1
Attaching Policies to WebLogic Web Services and Clients 17-2
Attaching Oracle WSM Policies to WebLogic Web Services 17-2
Attaching Oracle WSM Policies to WebLogic Web Service Clients 17-3
Attaching WebLogic Web Service Policies to WebLogic Web Services 17-3
Attaching WebLogic Web Service Policies to WebLogic Web Service Clients 17-3

Part V Reference

A Web Service Security Standards

Web Services Interoperability Organization—Basic Security Profile A-1
Transport Layer Security—SSL A-2
XML Encryption (Confidentiality) A-2
XML Signature (Integrity, Authenticity) A-3
WS-Security A-4
WS-Security Tokens A-4
Username A-4
X.509 Certificate A-5
Kerberos Ticket A-5
SAML Token A-5
WS-Policy A-7
WS-SecurityPolicy A-7
Web Services Addressing (WS-Addressing) A-8
WS-ReliableMessaging A-9

B Predefined Policies

Security Policies B-1
Authentication Only Policies B-1
oracle/wss_http_token_client_policy B-2
oracle/wss_http_token_service_policy B-2
oracle/wss_username_token_client_policy B-2
oracle/wss_username_token_service_policy B-3
oracle/wss10_saml_token_client_policy B-3
oracle/wss10_saml_token_service_policy B-3
oracle/wss11_kerberos_token_client_policy B-3
oracle/wss11_kerberos_token_service_policy B-4
Message Protection Only Policies B-4
oracle/wss10_message_protection_client_policy B-4

oracle/wss10_message_protection_service_policy B-4
oracle/wss11_message_protection_client_policy B-5
oracle/wss11_message_protection_service_policy B-5
Message Protection and Authentication Policies B-5
oracle/wss_http_token_over_ssl_client_policy B-7
oracle/wss_http_token_over_ssl_service_policy B-8
oracle/wss_saml_or_username_token_over_ssl_service_policy B-8
oracle/wss_saml_token_bearer_over_ssl_client_policy B-8
oracle/wss_saml_token_bearer_over_ssl_service_policy B-9
oracle/wss_saml_token_over_ssl_client_policy B-9
oracle/wss_saml_token_over_ssl_service_policy B-9
oracle/wss_username_token_over_ssl_client_policy B-9
oracle/wss_username_token_over_ssl_service_policy B-10
oracle/wss10_saml_hok_with_message_protection_client_policy B-10
oracle/wss10_saml_hok_token_with_message_protection_service_policy B-10
oracle/wss10_saml_token_with_message_integrity_client_policy B-11
oracle/wss10_saml_token_with_message_integrity_service_policy B-11
oracle/wss10_saml_token_with_message_protection_client_policy B-11
oracle/wss10_saml_token_with_message_protection_service_policy B-12
oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy B-12
oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy B-12
oracle/wss10_username_id_propagation_with_msg_protection_client_policy B-13
oracle/wss10_username_id_propagation_with_msg_protection_service_policy B-13
oracle/wss10_username_token_with_message_protection_client_policy B-14
oracle/wss10_username_token_with_message_protection_service_policy B-14
oracle/wss10_username_token_with_message_protection_
ski_basic256_client_policy B-15
oracle/wss10_username_token_with_message_protection_
ski_basic256_service_policy B-15
oracle/wss10_x509_token_with_message_protection_client_policy B-16
oracle/wss10_x509_token_with_message_protection_service_policy B-16
oracle/wss11_kerberos_token_with_message_protection_client_policy B-16
oracle/wss11_kerberos_token_with_message_protection_service_policy B-17
oracle/wss11_saml_token_with_message_protection_client_policy B-17
oracle/wss11_saml_token_with_message_protection_service_policy B-17
oracle/wss11_saml_or_username_token_with_message_protection_service_policy ... B-18
oracle/wss11_username_token_with_message_protection_client_policy B-18
oracle/wss11_username_token_with_message_protection_service_policy B-19
oracle/wss11_x509_token_with_message_protection_client_policy B-19
oracle/wss11_x509_token_with_message_protection_service_policy B-19
Authorization Only Policies B-20
oracle/binding_authorization_denyall_policy B-20
oracle/binding_authorization_permitall_policy B-21
oracle/binding_permission_authorization_policy B-21
oracle/component_authorization_denyall_policy B-21
oracle/component_authorization_permitall_policy B-21
oracle/component_permission_authorization_policy B-21

WS-Addressing Policies B-22
oracle/wsaddr_policy B-22
MTOM Attachment Policies B-22
oracle/wsmtom_policy B-22
Reliable Messaging Policies B-22
oracle/wsrm10_policy B-22
oracle/wsrm11_policy B-23
Management Policies B-23
oracle/log_policy B-23

C Predefined Assertion Templates

Security Assertion Templates C-1
Authentication Only Assertion Templates C-2
oracle/wss_http_token_client_template C-2
oracle/wss_http_token_service_template C-4
oracle/wss_username_token_client_template C-5
oracle/wss_username_token_service_template C-7
oracle/wss10_saml_token_client_template C-8
oracle/wss10_saml_token_service_template C-9
oracle/wss11_kerberos_token_client_template C-10
oracle/wss11_kerberos_token_service_template C-11
Message-Protection Only Assertion Template C-11
oracle/wss10_message_protection_client_template C-11
oracle/wss10_message_protection_service_template C-13
oracle/wss11_message_protection_client_template C-14
oracle/wss11_message_protection_service_template C-15
Message Protection and Authentication Assertion Templates C-16
oracle/wss_http_token_over_ssl_client_template C-17
oracle/wss_http_token_over_ssl_service_template C-19
oracle/wss_saml_token_bearer_over_ssl_client_template C-20
oracle/wss_saml_token_bearer_over_ssl_service_template C-21
oracle/wss_saml_token_over_ssl_client_template C-21
oracle/wss_saml_token_over_ssl_service_template C-21
oracle/wss_username_token_over_ssl_client_template C-21
oracle/wss_username_token_over_ssl_service_template C-23
oracle/wss10_saml_hok_with_message_protection_client_template C-24
oracle/wss10_saml_hok_with_message_protection_service_template C-27
oracle/wss10_saml_token_with_message_protection_client_template C-28
oracle/wss10_saml_token_with_message_protection_service_template C-30
oracle/wss10_username_token_with_message_protection_client_template C-31
oracle/wss10_username_token_with_message_protection_service_template C-34
oracle/wss10_x509_token_with_message_protection_client_template C-35
oracle/wss10_x509_token_with_message_protection_service_template C-37
oracle/wss11_kerberos_token_with_message_protection_client_template C-37
oracle/wss11_kerberos_token_with_message_protection_service_template C-39
oracle/wss11_saml_token_with_message_protection_client_template C-40
oracle/wss11_saml_token_with_message_protection_service_template C-42

oracle/wss11_username_token_with_message_protection_client_template C-43
oracle/wss11_username_token_with_message_protection_service_template C-46
oracle/wss11_x509_token_with_message_protection_client_template C-46
oracle/wss11_x509_token_with_message_protection_service_template....................... C-48
Authorization Assertion Templates C-49
oracle/binding_authorization_template C-49
oracle/binding_permission_authorization_template C-50
oracle/component_authorization_template C-51
oracle/component_permission_authorization_template C-52
Management Assertions.. C-53
oracle/security_log_template C-53
Supported Algorithm Suites C-54
Message Signing and Encyrption Settings for Request, Response, and Fault Messages C-54

D Schema Reference for Predefined Assertions

Graphical Representation.. D-1
Element Descriptions D-2
wsp:Policy D-3
Attributes D-3
Example.. D-3
wsp:ExactlyOne.. D-4
Attributes D-4
Example.. D-4
orasp:Assertion... D-5
Attributes D-5
Example.. D-6
orawsp:bindings... D-6
Example.. D-6
orawsp:Config D-6
Attributes D-6
Example.. D-7
orawsp:PropertySet.. D-7
Attributes D-7
Example.. D-7
orawsp:Property... D-7
Attributes D-8
Example.. D-10
orawsp:Description.. D-10
Example.. D-10
orawsp:Value D-10
Example.. D-10
oralgp:Logging D-10
Example.. D-10
orasp:binding-authorization.. D-11
Example.. D-11
orasp:binding-permission-authorization... D-11
Example.. D-11

orasp:coreid-security D-12
  Example D-12
orasp:http-security D-12
  Example D-13
orasp:kerberos-security D-13
  Example D-13
orasp:sca-component-authorization D-13
  Example D-14
orasp:sca-component-permission-authorization D-14
  Example D-14
orasp:wss10-anonymous-with-certificates D-15
  Example D-15
orasp:wss10-mutual-auth-with-certificates D-16
  Example D-16
orasp:wss10-saml-hok-with-certificates D-17
  Example D-17
orasp:wss10-saml-token D-18
  Example D-18
orasp:wss10-saml-with-certificates D-18
  Example D-18
orasp:wss10-username-with-certificates D-19
  Example D-19
orasp:wss11-anonymous-with-certificates D-20
  Example D-20
orasp:wss11-mutual-auth-with-certificates D-21
  Example D-21
orasp:wss11-saml-with-certificates D-22
  Example D-22
orasp:wss11-username-with-certificates D-23
  Example D-23
orasp:wss-saml-token-bearer-over-ssl D-24
  Example D-24
orasp:wss-saml-token-over-ssl D-25
  Example D-25
orasp:wss-username-token D-25
  Example D-26
orasp:wss-username-token-over-ssl D-26
  Example D-26
rm:RMAssertion D-27
  Example D-27
wsaw:UsingAddressing D-28
  Example D-28
wsoma:OptimizedMimeSerialization D-28
  Example D-28
oralgp:fault D-29
  Example D-29
oralgp:request D-29

Example D-29
oralgp:response D-29
Example D-29
oralgp:msg-log D-30
Example D-30
orasp:attachment D-30
Attributes D-30
Example D-30
orasp:auth-header D-30
Attributes D-30
Examples D-31
orasp:body D-31
Example D-31
orasp:check-permission D-31
Example D-31
orasp:coreid-token D-31
Attributes D-31
Example D-32
orasp:denyAll D-32
Example D-32
orasp:element D-32
Attributes D-32
Example D-32
orasp:encrypted-elements D-32
Example D-33
orasp:encrypted-parts D-33
Example D-33
orasp:fault D-33
Example D-33
orasp:header D-33
Attributes D-34
Example D-34
orasp:kerberos-token D-34
Attributes D-34
Example D-34
orasp:msg-security D-34
Attributes D-35
Example D-35
orasp:permitAll D-35
Example D-35
orasp:request D-36
Example D-36
orasp:require-tls D-36
Attributes D-36
Examples D-36
orawsp:resource-match D-36
Examples D-37

orasp:response D-37
Example D-37
orasp:role D-37
Attribute D-37
Example D-38
orasp:saml-token D-38
Attributes D-38
Example D-38
orasp:signed-elements D-38
Example D-38
orasp:signed-parts D-39
Example D-39
orasp:username-token D-39
Attributes D-39
Example D-40
orasp:x509-token D-40
Attributes D-40
Example D-40
orawsp:action-match D-41
Examples D-41
orawsp:Description D-41
Example D-41
orawsp:guard D-41
Examples D-41

E Schema Reference for Custom Assertions

Graphical Representation E-1
Element Descriptions E-1
wsp:Policy E-1
Attributes E-2
Example E-2
orasp:Assertion E-2
Attributes E-2
Example E-2
orawsp:bindings E-3
Example E-3
orawsp:Implementation E-3
Example E-3
orawsp:Config E-3
Attributes E-3
Example E-3
orawsp:PropertySet E-3
Attributes E-4
Example E-4
orawsp:Property E-4
Attributes E-4
Example E-4

orawsp:Description E-4
    Example E-4
orawsp:Value E-4
    Example E-4

Preface

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

About this Guide

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

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

Audience

This guide is intended for:

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

How to Use This Guide

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

The document is organized as follows:

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

  Chapter 4, "Examining the Rearchitecture of Oracle WSM in Oracle Fusion Middleware" discusses how the features of Oracle WSM have been rearchitected in Oracle Fusion Middleware 11*g* Release 1 (11.1.1). If you are an existing Oracle Web Services Manager 10*g* (Oracle WSM) customer, it is recommended that you review this chapter.

- Part II, "Basic Administration" describes the basic administration tasks that you can perform, such as deploying and configuring Web services; managing and attaching, and configuring policies; testing and monitoring Web services, and more.
- Part III, "Advanced Administration" describes the advanced administration tasks such as publishing and auditing Web services; migrating from a file-file-based store; creating custom assertions; managing policy lifecycle, diagnosing problems, interoperating with Oracle Fusion Middleware 11*g*, and more.
- Part IV, "WebLogic Web Service Administration" describes how to secure and administer WebLogic (Java EE) Web services.
- Part V, "Reference" provides reference information describing Web service security standards; predefined policy and assertion templates; and assertion schemas.

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at `http://www.fcc.gov/cgb/consumerfacts/trs.html`, and a list of phone numbers is available at `http://www.fcc.gov/cgb/dro/trsphonebk.html`.

Related Documents

For more information, see the following documents in the Oracle Fusion Middleware 11*g* Release 1 (11.1.1) documentation set:

- *Oracle Fusion Middleware Introducing Web Services*
- *Oracle Fusion Middleware Concepts Guide for Oracle Infrastructure Web Services*
- *Oracle Fusion Middleware Introducing WebLogic Web Services for Oracle WebLogic Server*
- *Oracle Fusion Middleware Getting Started With JAX-WS Web Services for Oracle WebLogic Server*
- *Oracle Fusion Middleware Programming Advanced Features of JAX-WS Web Services for Oracle WebLogic Server*
- *Oracle Fusion Middleware Getting Started With JAX-RPC Web Services for Oracle WebLogic Server*
- *Oracle Fusion Middleware Programming Advanced Features of JAX-RPC Web Services for Oracle WebLogic Server*
- *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server*
- *Oracle Fusion Middleware WebLogic Web Services Reference for Oracle WebLogic Server*
- *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*
- *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*
- *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter*
- "Developing with Web Services" in the Oracle JDeveloper online help

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

Conventions

The following text conventions are used in this document:

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

What's New

11*g* Release 1 (11.1.1) includes a complete redesign of Oracle Web Services Manager 10*g* and Web services security management. For more details about what has changed in Release 11*g*, see Chapter 4, "Examining the Rearchitecture of Oracle WSM in Oracle Fusion Middleware."

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

- Enhanced administration and policy management for asynchronous Web services
- Ability to define policy alternatives (OR groups)
- Service-side policy configuration overrides
- Oracle WSM policy attachment using the WebLogic Scripting Tool (WLST)
- Ability to upgrade the Oracle WSM policies in the Metadata Services (MDS) repository using WLST commands
- Service identity certification extension for Web services that implement a message-protection policy. The Web service's public certificate is published in the WSDL, and it is no longer necessary for the Web service client to store the Web service's public certificate in its domain-level keystore.
- Enhanced support for permission-based authorization using the oracle.wsm.security.WSFunctionPermission permission check class. In this release, the resource target of the WSFunctionPermission is enhanced to include the actual Web service operation name.
- Ability to browse WSIL documents and import UDDI v3 registries using Fusion Middleware Control, and register services accordingly
- Compliance with WSI-Basic Security Profile
- Support for testing RESTful Web services in Fusion Middleware Control Test Web Service page
- Support for Microsoft SQL Server in the MDS repository
- Ability to use the same MDS repository to manage policies across multiple domains. In previous releases, an MDS repository could only be used by a single domain.
- New document, *Oracle Fusion Middleware Interoperability Guide for Oracle Web Services Manager*, that contains the interoperability content previously provided in this document
- Interoperability is certified between Oracle Web Services Manager and Axis 1.4 and WSS4J 1.58 security environments

11*g* Release 1 (11.1.1) includes the following new features:

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

Part I

Introduction

Part I contains the following chapters:

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

1 Overview of Web Services Security and Administration

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

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

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

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

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

- **Oracle Web Services Manager (WSM) security and management has been completely redesigned and rearchitected.** The previous release, Oracle WSM 10*g*, was delivered as a standalone product or as a component of the Oracle SOA Suite. In the 11*g* release, Oracle WSM has been integrated into the Oracle WebLogic Server. For complete details, see "Examining the Rearchitecture of Oracle WSM in Oracle Fusion Middleware" on page 4-1.
- **Oracle Web services can be classified into the following categories**:
  - WebLogic (Java EE) Web services (see "Securing and Administering WebLogic Web Services" on page 1-4)
  - Oracle Infrastructure Web services—SOA, ADF, and WebCenter services (see "Securing and Administering Oracle Infrastructure Web Services" on page 1-3)

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

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

Table 1–1 Types of Web Service Policies

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

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

Web Service Security and Administration Tasks

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

- Deploy, configure, test, and monitor Web services.
- Enable, publish, and register Web services.
- Attach policies to secure and manage Web services and analyze policy usage.
- Create new policies and assertion templates, and manage and configure existing policies.
- Create custom assertions to meet the requirements of your application.
- Manage policy lifecycle to transition from a test to production environment.
- Manage your file-based and database stores in your development and production environments, respectively.
- Test interoperability with other Web services.
- Diagnose problems.

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

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

Securing and Administering Oracle Infrastructure Web Services

To secure and administer Oracle Infrastructure Web services:

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

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

Use this tool...	To...
Oracle Enterprise Manager Fusion Middleware Control	Secure and administer SOA, ADF, and WebCenter services, performing the tasks described in "Web Service Security and Administration Tasks" on page 1-2. To access Oracle Enterprise Manager Fusion Middleware Control, see "Accessing Oracle Enterprise Manager Fusion Middleware Control" on page 1-5. Oracle Enterprise Manager Fusion Middleware Control leverages Oracle Web Services Manager (WSM) to centrally define security and management policies, and enforce them locally at runtime. For more information about Oracle WSM, see "Understanding Oracle WSM Policy Framework" on page 3-1. For more information about Oracle Enterprise Manager Fusion Middleware Control, see "Getting Started Using Oracle Enterprise Manager Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide*.
WebLogic Scripting Tool (WLST)	Perform Web service configuration and policy management tasks. To access WLST, see "Accessing the Web Services Custom WLST Commands" on page 1-6. For more information about using WLST, see "Getting Started Using the Oracle WebLogic Scripting Tool (WLST)" in *Oracle Fusion Middleware Administrator's Guide*.

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

Securing and Administering WebLogic Web Services

To secure and administer WebLogic Web services:

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

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

Use this tool . . .	To perform the following tasks . . .
Oracle Enterprise Manager Fusion Middleware Control	Leverage Oracle WSM to perform the following tasks: ■ Enforce policies at runtime. ■ Test the WebLogic Web service. ■ Monitor the performance of WebLogic Web services. For more information about Oracle WSM, see "Understanding Oracle WSM Policy Framework" on page 3-1. To access Oracle Enterprise Manager Fusion Middleware Control, see "Accessing Oracle Enterprise Manager Fusion Middleware Control" on page 1-5. For more information about Oracle Enterprise Manager Fusion Middleware Control, see "Getting Started Using Oracle Enterprise Manager Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide*. **Note**: The following features are *not supported* for WebLogic Web services in the 11*g* release: ■ Centralized policy management of Oracle WSM policies. ■ Ability to advertise policies. ■ WS-SecureConversation, WS-Trust, MTOM, WS-Addressing, WS-ReliableMessaging, or WS-AtomicTransaction policies. ■ Security and administration of JAX-RPC WebLogic Web services.
Oracle WebLogic Server Administration Console	Perform all of the tasks described in "Web Service Security and Administration Tasks" on page 1-2 to secure and manage WebLogic Web services. To access Oracle WebLogic Server Administration Console, see "Accessing Oracle WebLogic Administration Console" on page 1-6. For more information about using the Oracle WebLogic Server Administration Console to secure and administer WebLogic Web services, see "Web Services" in the *Oracle WebLogic Server Administration Console Online Help*.

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

Accessing the Security and Administration Tools

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

Accessing Oracle Enterprise Manager Fusion Middleware Control

To access Oracle Enterprise Manager Fusion Middleware Control:

1. Start the Oracle WebLogic Server.

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

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

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

   The Login page displays.

3. Enter the username and password.

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

4. Click **Login**.

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

Accessing Oracle WebLogic Administration Console

To access Oracle WebLogic Administration Console:

1. Start the Oracle WebLogic Server.

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

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

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

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

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

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

   The Login page displays.

3. Enter the username and password.

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

4. Click **Log In**.

For more information, see "Starting the Console" in the *Oracle WebLogic Administration Console Online Help*.

Accessing the Web Services Custom WLST Commands

To access the Web services WLST commands:

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

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

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

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

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

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

3. Start Oracle WebLogic Server.

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

4. Connect to the running WebLogic Server instance using the `connect()` command. For example, the following command connects WLST to the Admin Server at the URL `myAdminServer.oracle.com:7001` using the username/password credentials `weblogic/welcome1`:

   ```
   connect("weblogic","welcome1","t3://myAdminServer.oracle.com:7001")
   ```

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

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

2 Understanding Web Services Security Concepts

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

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

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

Securing Web Services

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

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

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

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

- **Authorization (or Access Control)**—Granting access to specific resources based on an authenticated user's entitlements. Entitlements are defined by one or several attributes. An attribute is the property or characteristic of a user, for example, if "Marc" is the user, "conference speaker" is the attribute.

- **Confidentiality, privacy**—Keeping information secret. Accesses a message, for example a Web service request or an email, as well as the identity of the sending and receiving parties in a confidential manner. Confidentiality and privacy can be achieved by encrypting the content of a message and obfuscating the sending and receiving parties' identities.

- **Integrity, non repudiation**—Making sure that a message remains unaltered during transit by having the sender digitally sign the message. A digital signature is used to validate the signature and provides non-repudiation. The timestamp in the signature prevents anyone from replaying this message after the expiration.

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

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

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

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

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

Transport-level Security

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

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

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

Application-level Security

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

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

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

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

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

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

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

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

Web Service Security Requirements

The following summarize the Web service security requirements:

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

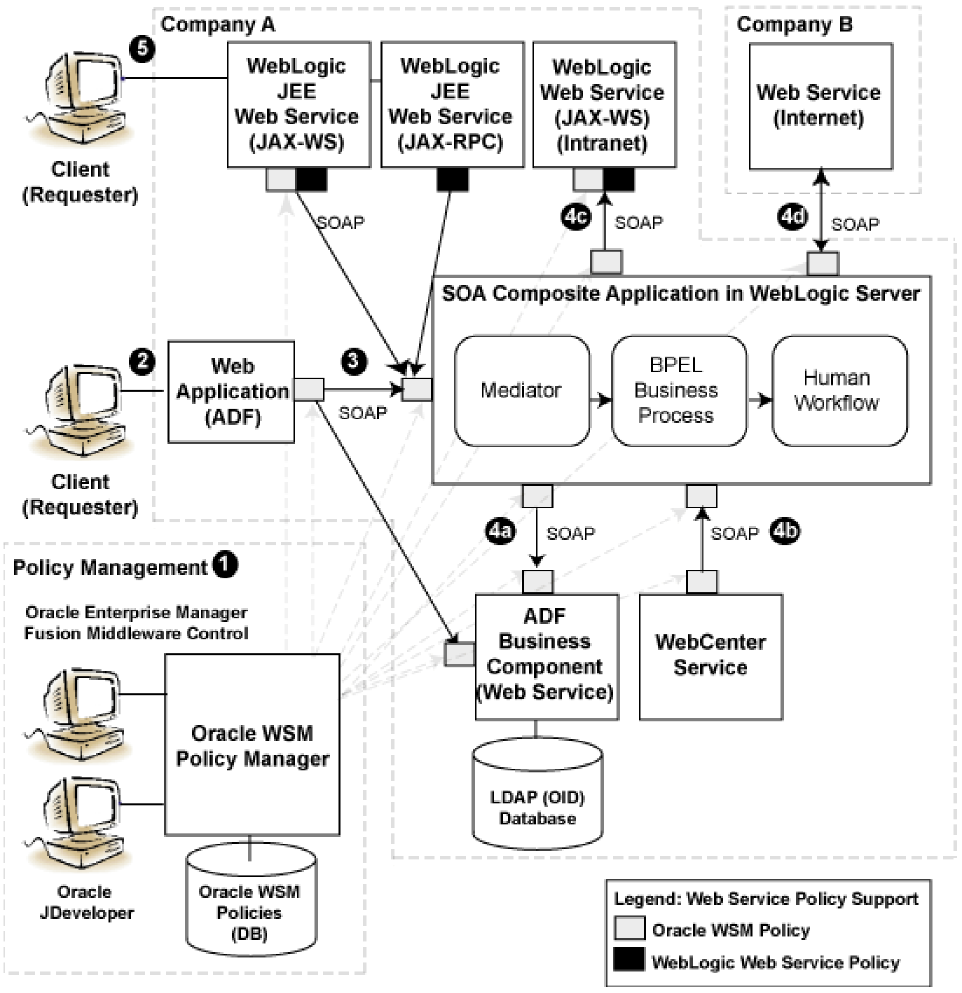
Oracle Web Services Manager (WSM) is designed to define and implement Web services security in heterogeneous environments, including authentication, authorization, message encryption and decryption, signature generation and validation, and identity propagation across multiple Web services used to complete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

How Oracle Fusion Middleware Secures Web Services and Clients

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

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

Figure 2–1 Example of Oracle Fusion Middleware Application


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

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

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

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

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

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

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

   The user may be internal or external to Company A.

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

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

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

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

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

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

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

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

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

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

5. A client accesses a WebLogic JEE Web service.

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

3 Understanding Oracle WSM Policy Framework

This chapter contains the following sections:

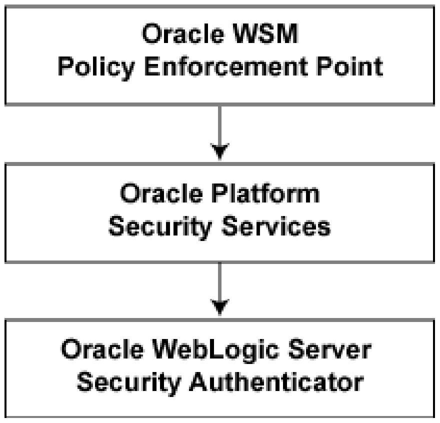
- Overview of Oracle WSM Policy Framework
- What Are Policies?
- Building Policies Using Policy Assertions
- Attaching Policies to Subjects
- How Policies are Executed
- Oracle WSM Predefined Policies and Assertion Templates
- Defining Multiple Policy Alternatives (OR Groups)
- Overriding Security Policy Configuration
- Recommended Naming Conventions for Policies

Overview of Oracle WSM Policy Framework

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

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

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



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

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

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

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

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

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

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

Figure 3–2 Components of Oracle WSM Architecture

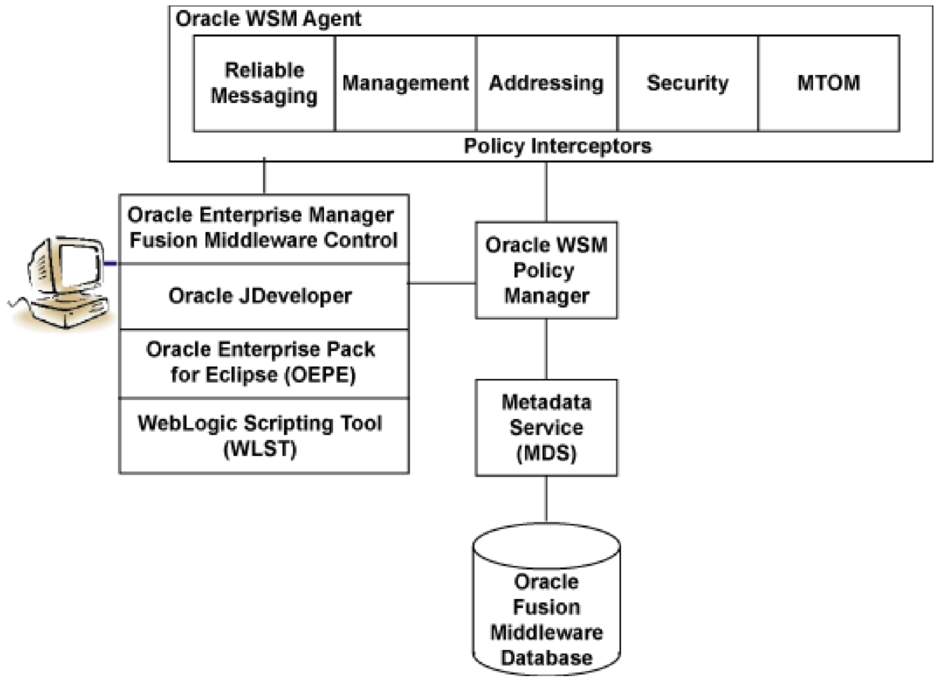

Oracle WSM Agent
Reliable Messaging
Management
Addressing
Security
MTOM
Policy Interceptors
Oracle Enterprise Manager Fusion Middleware Control
Oracle JDeveloper
Oracle Enterprise Pack for Eclipse (OEPE)
WebLogic Scripting Tool (WLST)
Oracle WSM Policy Manager
Metadata Service (MDS)
Oracle Fusion Middleware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

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

Table 3–1 Components of Oracle WSM Architecture

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

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

Oracle WSM Component	Description
Policy Interceptors	Enforce policies, including reliable messaging, management, addressing, security, and Message Transmission Optimization Mechanism (MTOM). For more information, see "How Policies are Executed" on page 3-6.
Metadata Service (MDS)	Stores policies. Policies can be stored either as files in the file system (supported for development) or to the Oracle Fusion Middleware database (supported for production).
Oracle Fusion Middleware Database	Provides database support for the MDS.

What Are Policies?

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

Oracle Fusion Middleware 11*g* Release 1 (11.1.1) supports the types of policies defined in Table 3–2. The policies are part of the Oracle WSM enterprise policy framework which allows policies to be centrally created and managed.

Table 3–2 Types of Policies

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

Table 3–2 (Cont.) Types of Policies

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

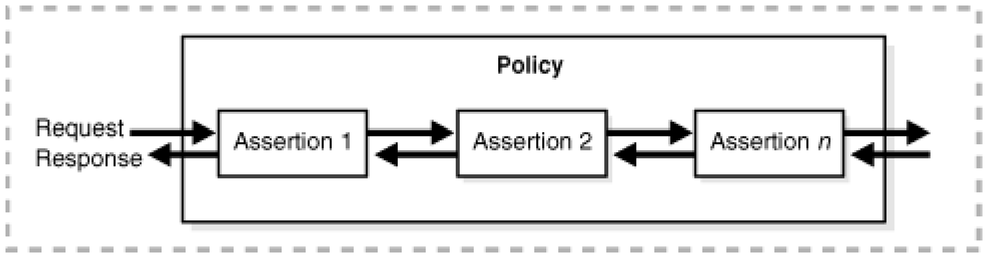
Building Policies Using Policy Assertions

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

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

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

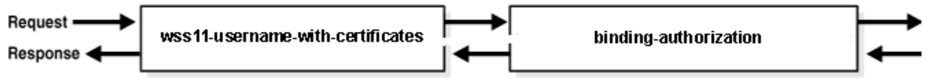
Figure 3–3 Policy Containing Assertions



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

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

Figure 3–4   Example Policy With Two Assertions



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

Attaching Policies to Subjects

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

You can attach one or more policies to a policy subject, either individually or as a bulk attachment. When the policy is attached to a policy subject, enforcement of the policy begins immediately.

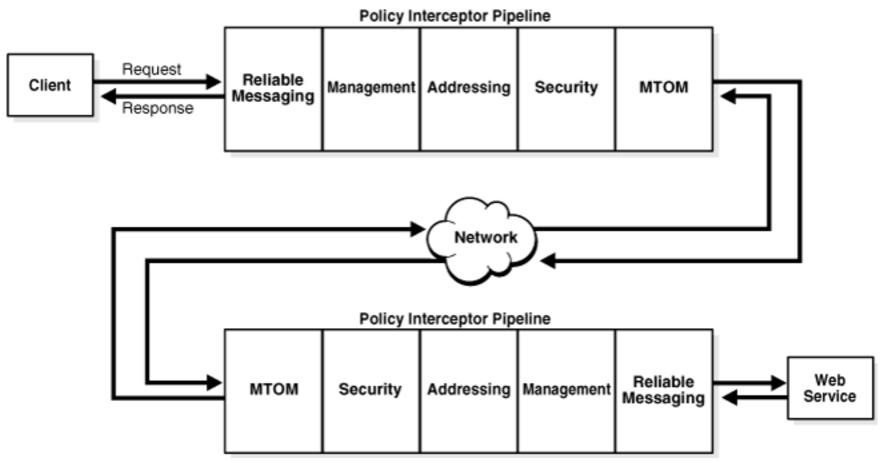
If a policy on the client side is modifying the message, for example to encrypt the message, there must be a corresponding policy on the Web service side, for example, to decrypt the policy. Otherwise, the message request will fail.

How Policies are Executed

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

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

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



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

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

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

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

Oracle WSM Predefined Policies and Assertion Templates

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

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

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

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

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

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

Defining Multiple Policy Alternatives (OR Groups)

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

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

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

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

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

There are two predefined service policies that contain OR groups:

- oracle/wss_saml_or_username_token_over_ssl_service_policy—For more information, see "oracle/wss_saml_or_username_token_over_ssl_service_policy" on page B-8.
- oracle/wss11_saml_or_username_token_with_message_protection_service_policy—For more information, see "oracle/wss11_saml_or_username_token_with_message_protection_service_policy" on page B-18.

Overriding Security Policy Configuration

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

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

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

For more information about overriding security policy configuration, see "Attaching Client Policies Permitting Overrides" on page 8-13 and "Attaching Web Service Policies Permitting Overrides" on page 8-14.

You can define whether a configuration property is overridable when creating custom assertions, as described in "Creating Custom Assertions" on page 14-1.

Recommended Naming Conventions for Policies

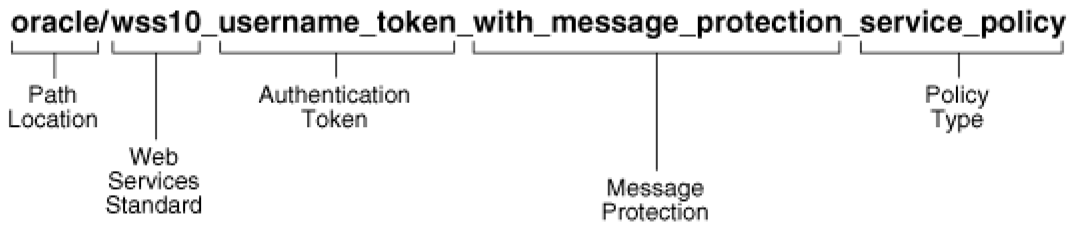
The valid characters for directory, policy, and assertion template names are:

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

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

Oracle recommends that you encode as much information as possible into the name of the policy so that you can tell, at a glance, what the policy does. For example, one of the predefined security policies that is delivered with Oracle Fusion Middleware 11*g* Release 1 (11.1.1) is named oracle/wss10_username_token_with_message_protection_service_policy. Figure 3–6 identifies the different parts of this predefined policy name.

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



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

- Path Location – All policies are identified by the directory in which the policy is located. All predefined policies that come with the product are in the `oracle` directory.
- Web services Standard – If the policy uses a WS-Security standard, it is identified with wss10 (WS-Security 1.0) or wss11 (WS-Security 1.1). Or it could just be set to wss to indicate that it is independent of WS-Security 1.0 or 1.1.
- Authentication token – If the policy authenticates users, then the type of token is specified. The predefined options include:
  - http_token – HTTP token
  - kerberos_token – Kerberos token
  - saml_token – SAML token
  - username_token – Username and password token
  - x509_token – X.509 certificate token

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

  wss10_message_protection_service_policy

  wss10_message_protection_service_template

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

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

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

Note: You cannot prefix the name of a policy with "oracle_". For example, oracle_wss_http_token_service_policy is not a valid policy name.

4

Examining the Rearchitecture of Oracle WSM in Oracle Fusion Middleware

In Oracle Fusion Middleware 11*g* Release 1 (11.1.1), Oracle Web Services Manager (WSM) security and management has been completely redesigned and rearchitected. The previous release, Oracle WSM 10*g*, was delivered as a standalone product or as a component of the Oracle SOA Suite. In the 11*g* release, Oracle WSM has been integrated with Oracle WebLogic Server as part of the Oracle Fusion Middleware SOA Suite.

This chapter contains the following sections:

- How Oracle WSM 10g is Redesigned in Oracle Fusion Middleware 11g Release 1 (11.1.1)
- Comparing Oracle WSM 10g and Oracle WSM 11g Policies
- Comparing Oracle Application Server 10g WS-Security with Oracle WSM 11g
- Interoperability and Upgrade

How Oracle WSM 10*g* is Redesigned in Oracle Fusion Middleware 11*g* Release 1 (11.1.1)

Oracle WSM 10*g* has been rearchitected in Oracle Fusion Middleware 11*g* Release 1 (11.1.1), as follows:

- **Oracle WSM Agent functionality is integrated into Oracle WebLogic Server.** In Oracle Fusion Middleware 11g, the Oracle WSM 10*g* Agents are managed by the security and management policy interceptors.
- **Policy management and monitoring is integrated into Oracle Enterprise Manager Fusion Middleware Control.** The functions of the Oracle WSM Monitor and the Web Services Manager Control have been integrated into Fusion Middleware Control. This allows you to manage your enterprise from one central location.
- **Oracle WSM Policy Manager enforces additional Web service QoS requirements.** The Oracle WSM Policy Manager manages not only security policies, but it also manages other types of policies such as Message Transmission Optimization Mechanism (MTOM), Reliable Messaging, Addressing, and Management.
- **The Oracle WSM Database is replaced by the Oracle Metadata Repository and Oracle Fusion Middleware Database.** The database continues to store policies and monitoring data in 11*g*. MDS provides integration with a common Metadata Repository.

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

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

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

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

- Oracle WSM 10*g* supported policy enforcement agents for third-party application servers, such as IBM WebSphere and Red Hat JBoss. Oracle Fusion Middleware 11*g* Release 1 (11.1.1) only supports Oracle WebLogic Server. Support for third-party application servers will follow this release.

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

Table 4–1 Comparison of Oracle WSM 10g and Oracle Fusion Middleware 11g Release 1 (11.1.1)

	Description of Functionality	**Oracle WSM 10*g* Component**	**Oracle Fusion Middleware 11*g* Release 1 (11.1.1) Component**
1	Policy enforcement point	Oracle WSM Server and Client Agents, Oracle WSM Gateway	Oracle WSM Agent which manages the policy interceptors There is no equivalent component for the Oracle WSM Gateway in Oracle Fusion Middleware 11*g* Release 1 (11.1.1).
2	GUI Component to author policies and attach policies to Web services	Web Services Manager Control	Oracle Enterprise Manager Fusion Middleware Control
3	Component to manage policies	Oracle WSM Policy Manager	Oracle WSM Policy Manager
4	Component used to monitor Web services data	Oracle WSM Monitor	Oracle Enterprise Manager Fusion Middleware Control and Oracle Enterprise Manager Grid Control
5	Policy Store	Oracle WSM Database	Oracle Metadata Services (MDS) Repository

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

Figure 4–1 Oracle WSM 10g Components

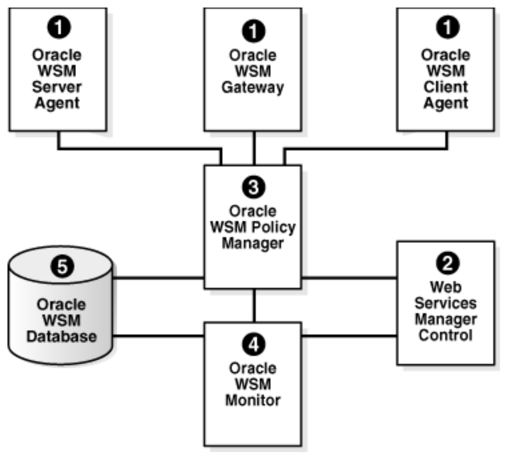
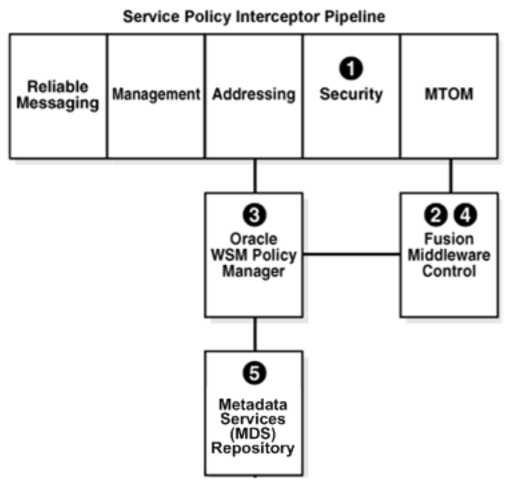

1 Oracle WSM Server Agent
1 Oracle WSM Gateway
1 Oracle WSM Client Agent
3 Oracle WSM Policy Manager
5 Oracle WSM Database
2 Web Services Manager Control
4 Oracle WSM Monitor

Figure 4–2 shows the Oracle Fusion Middleware 11*g* Release 1 (11.1.1) components, and the numbers in Table 4–1 correspond to the components in the figure.

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

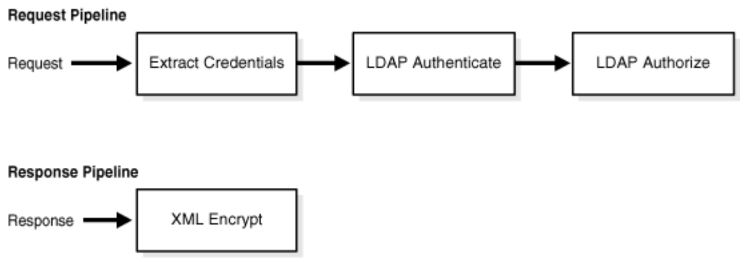

Service Policy Interceptor Pipeline
Reliable Messaging
Management
Addressing
1 Security
MTOM
3 Oracle WSM Policy Manager
2 4 Fusion Middleware Control
5 Metadata Services (MDS) Repository

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

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

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

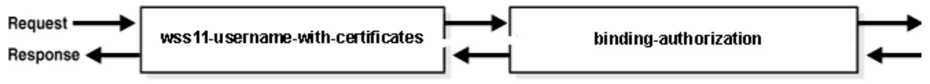
Figure 4–3 Oracle WSM 10g Policy Pipeline



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

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

Figure 4–4 Oracle WSM 11g Policy Pipeline



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

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

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

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

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

Interoperability and Upgrade

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

- Oracle WSM, as described in "Interoperability with Oracle WSM 10g Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.
- Oracle WSM gateways, as described in "Interoperability with Oracle WSM 10g Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.
- Application Server, as described in "Interoperability with Oracle Containers for J2EE (OC4J) 10g Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.

In addition, you can interoperate with the following components:

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

You can upgrade the following 10.1.3 features to Oracle Fusion Middleware 11*g* Release 1 (11.1.1):

- OC4J Web services 10.1.3 to WebLogic Web services. See "Upgrading Your Java EE Applications" in *Upgrade Guide for Java EE Release 11g*.
- Oracle WSM 10.1.3 policies to Oracle WSM 11g . See "Upgrading Oracle Web Services Manager (WSM) Policies" in *Upgrade Guide for Oracle SOA Suite, WebCenter, and ADF Release 11g*.
- Oracle Containers for Java (OC4J) 10.1.3 security environments to OWSM 11g. See "Upgrading Oracle Containers for J2EE (OC4J) Security Environments" in *Upgrade Guide for Oracle SOA Suite, WebCenter, and ADF Release 11g*.

Part II

Basic Administration

Note: For information about securing and administering WebLogic Web services, see Chapter 17, "Securing and Administering WebLogic Web Services."

Part II contains the following chapters:

- Chapter 5, "Deploying Web Services Applications"
- Chapter 6, "Administering Web Services"
- Chapter 7, "Managing Web Service Policies"
- Chapter 8, "Attaching Policies to Web Services"
- Chapter 9, "Setting Up Your Environment for Policies"
- Chapter 10, "Configuring Policies"
- Chapter 11, "Testing Web Services"
- Chapter 12, "Monitoring the Performance of Web Services"

5 Deploying Web Services Applications

This chapter contains the following sections:

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

Overview

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

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

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

Additional Deployment Documentation Available

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

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

- *Deploying, Undeploying, and Redeploying WebCenter Applications*

Deploying Web Services Applications

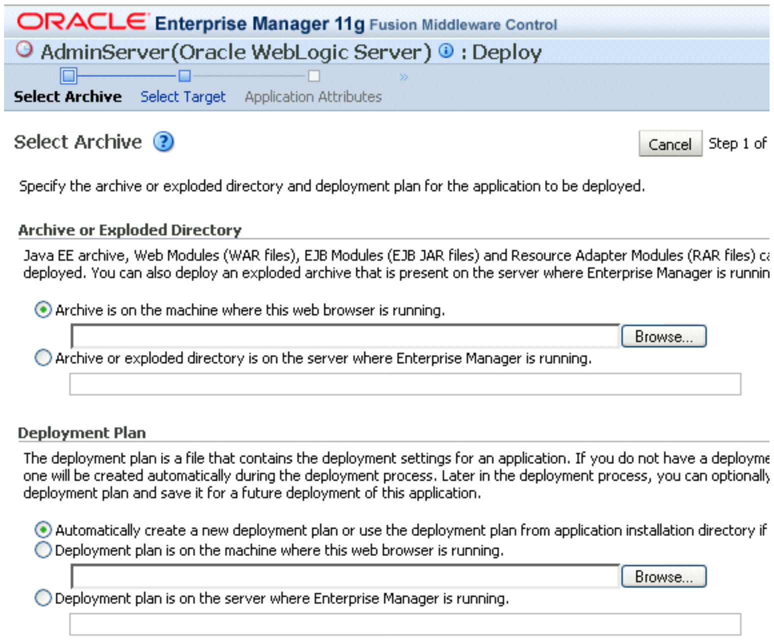
The following is an overview of the basic procedure for deploying a Web service application using the Oracle Enterprise Manager Fusion Middleware Control.

To deploy a Web services application

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

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

Figure 5–1 Select Archive Page


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

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

Archive or Exploded Directory

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

[✓ Archive is on the machine where this web browser is running.]

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

Deployment Plan

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

[✓ Automatically create a new deployment plan or use the deployment plan from application installation directory if]

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

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

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

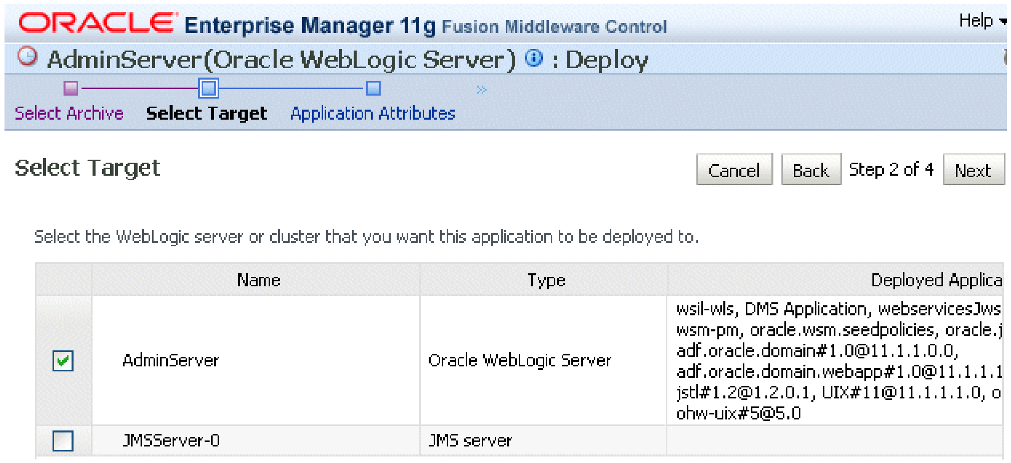
   Click one of the following Deployment Plan options:
   - Automatically create a new deployment plan
   - Deployment plan is present on local host

- Deployment plan is already present on the server where Enterprise Manager is running

7. Click **Next**.

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

Figure 5–2 Select Target Page


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

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

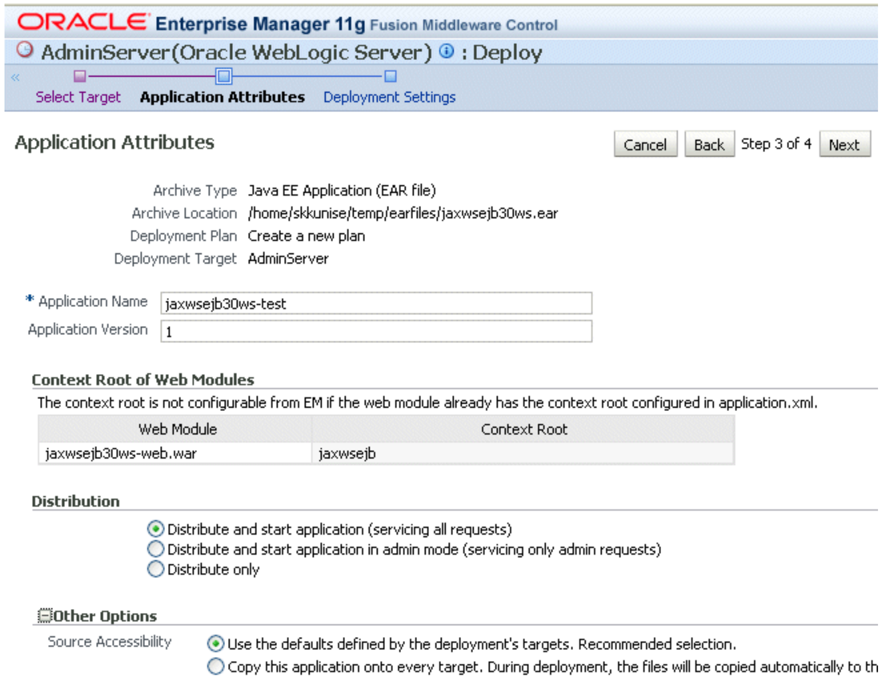
	Name	Type	Deployed Applica
✓	AdminServer	Oracle WebLogic Server	wsil-wls, DMS Application, webservicesJws wsm-pm, oracle.wsm.seedpolicies, oracle.j adf.oracle.domain#1.0@11.1.1.0.0, adf.oracle.domain.webapp#1.0@11.1.1.1 jstl#1.2@1.2.0.1, UIX#11@11.1.1.1.0, o ohw-uix#5@5.0
	JMSServer-0	JMS server	

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

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

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

Figure 5–3 Application Attributes Page



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

* Application Name: jaxwsejb30ws-test

Application Version: 1

Context Root of Web Modules

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

Web Module	Context Root
jaxwsejb30ws-web.war	jaxwsejb

Distribution

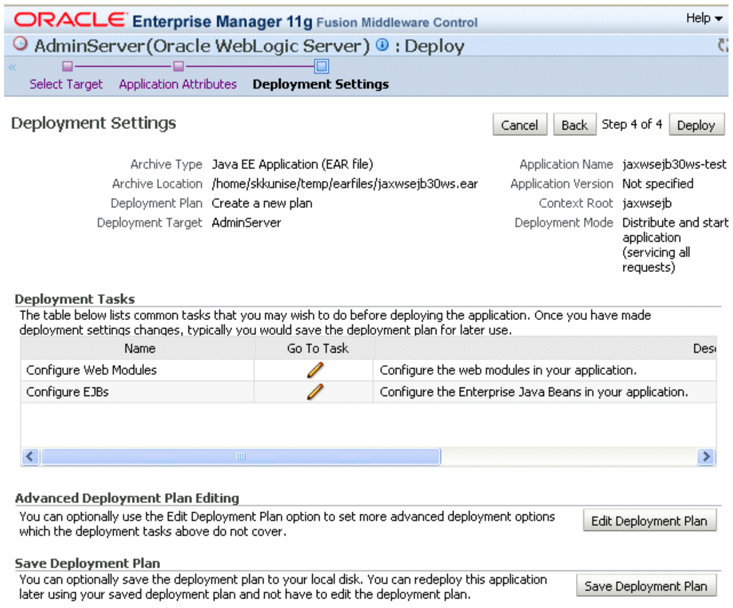
[✓ Distribute and start application (servicing all requests)]
Distribute and start application in admin mode (servicing only admin requests)
Distribute only

Other Options

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

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

Figure 5–4 Deployment Settings Page



Deployment Settings

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

Deployment Tasks

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

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

Advanced Deployment Plan Editing

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

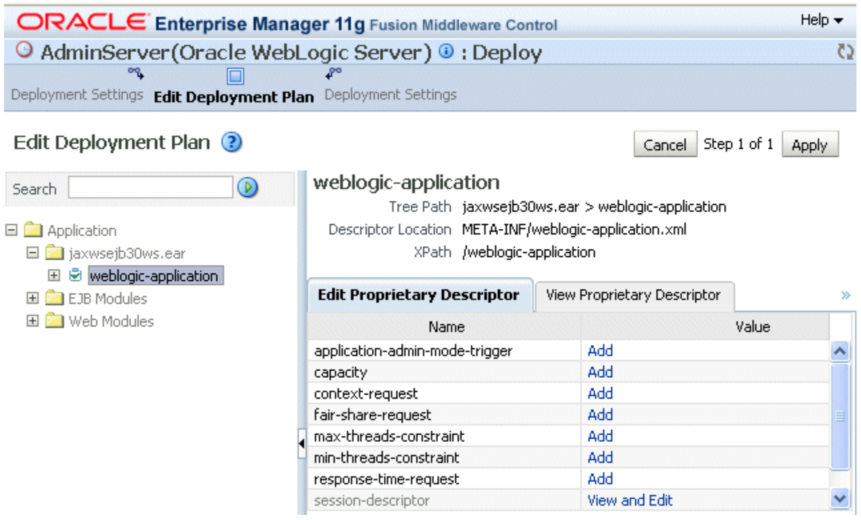
Save Deployment Plan

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

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

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

Figure 5–5 Edit Deployment Plan



weblogic-application

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

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

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

Undeploying a Web Services Application

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

To undeploy a Web services application

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

   The Application Deployment is displayed

2. Using Fusion Middleware Control, click **Application Deployment**.
3. From the **Application Deployment** menu, select **Application Deployment**, then **Undeploy**.

   The undeploy confirmation page is displayed.

4. Click **Undeploy**.

   Processing messages are displayed.

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

Redeploying a Web Services Application

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

Redeploy an application if:

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

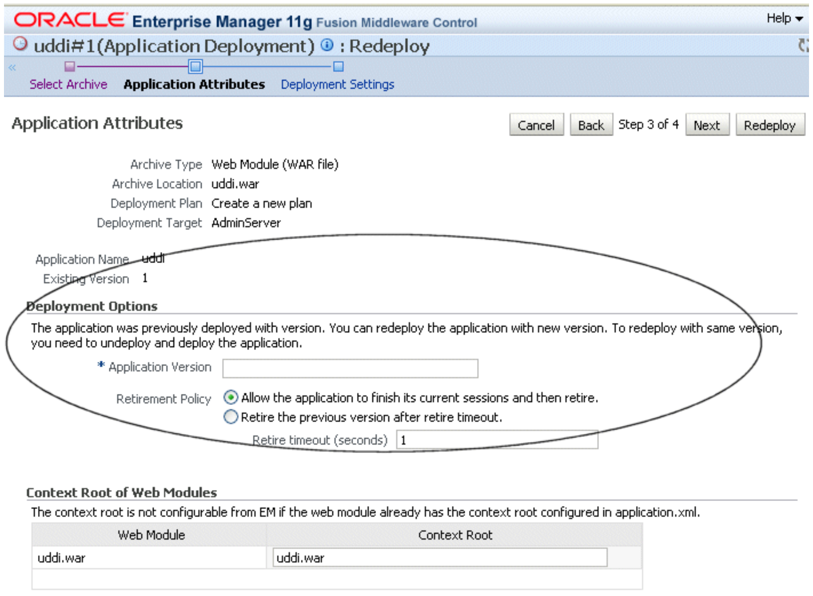
When you redeploy an application, you can redeploy the original archive file or exploded directory, or you can specify a new archive file in place of the original one. You can also change the deployment plan that is associated with the application.

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

To redeploy a Web services application

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

Figure 5–6 Setting Application Attributes During Redeploy


ORACLE Enterprise Manager 11g Fusion Middleware Control
Help
uddi#1(Application Deployment) : Redeploy
Select Archive
Application Attributes
Deployment Settings
Cancel
Back
Step 3 of 4
Next
Redeploy

Application Attributes

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

Application Name	uddi
Existing Version	1

Deployment Options

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

* Application Version

Retirement Policy [✓ Allow the application to finish its current sessions and then retire.] Retire the previous version after retire timeout.

Retire timeout (seconds) 1

Context Root of Web Modules

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

Web Module	Context Root
uddi.war	uddi.war

6 Administering Web Services

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

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

Note: As described in Chapter 17, "Securing and Administering WebLogic Web Services", you use Oracle Enterprise Manager Fusion Middleware Control to test and monitor Java EE Web services. For all other configuration tasks you use the WebLogic Server Administration Console.

The Web services pages described in this chapter have different content for Java EE, ADF and WebCenter Web and SOA services. The pages for ADF and WebCenter and SOA Web services are shown in the figures.

Viewing All Current Web Services for a Server

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

Using Fusion Middleware Control

To view all current Web services for a server:

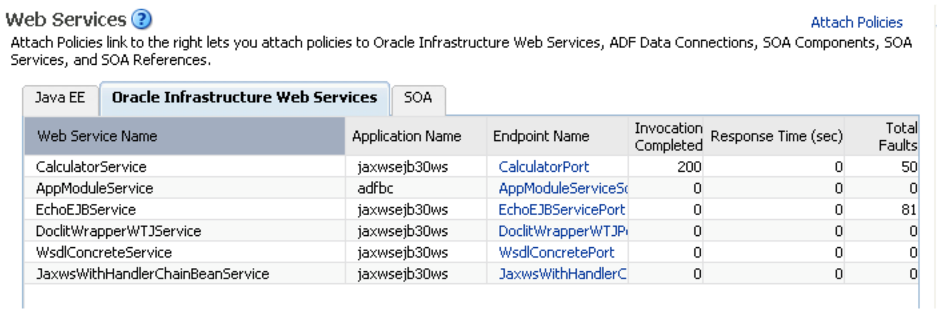
1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to see the Web services.
2. Expand the domain.
3. Select the server for which you want to view all current Web services.
4. Using Fusion Middleware Control, click **WebLogic Server** and then **Web Services**. The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

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

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

   For ADF and WebCenter and SOA Services, from this page you can click **Attach Policies** to attach one or more policies to one or more Web services.

Figure 6–1 Server-Specific Web Services Summary Page



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

Web Service Name	Application Name	Endpoint Name	Invocation Completed	Response Time (sec)	Total Faults
CalculatorService	jaxwsejb30ws	CalculatorPort	200	0	50
AppModuleService	adfbc	AppModuleServiceS	0	0	0
EchoEJBService	jaxwsejb30ws	EchoEJBServicePort	0	0	81
DoclitWrapperWTJService	jaxwsejb30ws	DoclitWrapperWTJP	0	0	0
WsdlConcreteService	jaxwsejb30ws	WsdlConcretePort	0	0	0
JaxwsWithHandlerChainBeanService	jaxwsejb30ws	JaxwsWithHandlerC	0	0	0

Using WLST

To view all the current Web services for a server:

1. Connect to the running instance of WebLogic Server for which you want to view the Web services as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.

2. Use the `listWebServices()` WLST command to display a list of the Web services.

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

   For example:

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

   moduleName=jaxwsejb,moduleType=web,serviceName=JaxwsWithHandlerChainBeanService
    moduleName=jaxwsejb, moduleType=web, serviceName=WsdlConcreteService
    moduleName=jaxwsejb, moduleType=web, serviceName=EchoEJBService
   ```

3. Set the `detail` argument of the `listWebServices` command to `true` to view port and policy information for the Web services.

   For example:

   ```
   wls:/wls-domain/serverConfig> listWebServices(None,None,true)

   /wls-domain/AdminServer/jaxwsejb30ws :
           moduleName=jaxwsejb, moduleType=web,
   serviceName=JaxwsWithHandlerChainBeanService
           enableTestPage: true
           enableWSDL: true

                   JaxwsWithHandlerChainBeanPort
   http://host.us.oracle.com:7001/jaxwsejb/JaxwsWithHandlerChainIntf
                   enable: true
                   enableREST: false
                   maxRequestSize: -1
                   loggingLevel: NULL

           moduleName=jaxwsejb, moduleType=web, serviceName=WsdlConcreteService
           enableTestPage: true
           enableWSDL: true

                   WsdlConcretePort
   http://host.us.oracle.com:7001/jaxwsejb/WsdlAbstract
                   enable: true
                   enableREST: false
                   maxRequestSize: -1
                   loggingLevel: NULL
           moduleName=jaxwsejb, moduleType=web, serviceName=EchoEJBService
           enableTestPage: true
           enableWSDL: true

                   EchoEJBServicePort
   http://host.us.oracle.com:7001/jaxwsejb/EchoEJBService
                   enable: true
                   enableREST: false
                   maxRequestSize: -1
                   loggingLevel: NULL
   ```

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

Navigating to the Web Services Summary Page for an Application

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

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

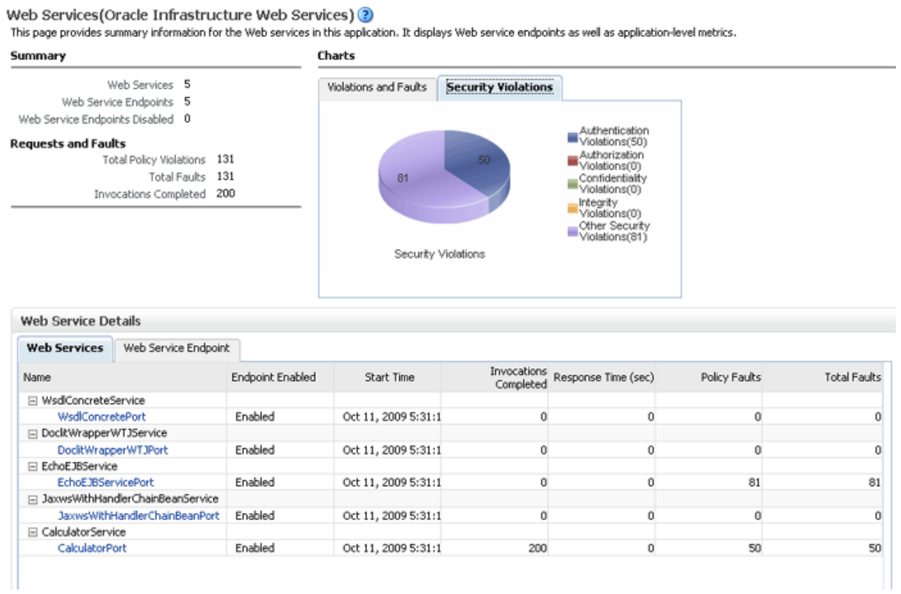
1. From the navigator pane, click the plus sign (+) for the Application Deployments folder to expose the applications in the farm, and select the application.

   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 (Figure 6–2) for your application.

Figure 6–2 Web Services Home Page



Web Services(Oracle Infrastructure Web Services)

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

Summary

Web Services	5
Web Service Endpoints	5
Web Service Endpoints Disabled	0

Requests and Faults

Total Policy Violations	131
Total Faults	131
Invocations Completed	200

Charts
Violations and Faults
Security Violations
50
81
Authentication Violations(50)
Authorization Violations(0)
Confidentiality Violations(0)
Integrity Violations(0)
Other Security Violations(81)
Security Violations

Web Service Details

Web Services | Web Service Endpoint

Name	Endpoint Enabled	Start Time	Invocations Completed	Response Time (sec)	Policy Faults	Total Faults
⊟ WsdlConcreteService						
WsdlConcretePort	Enabled	Oct 11, 2009 5:31:1	0	0	0	0
⊟ DoclitWrapperWTJService						
DoclitWrapperWTJPort	Enabled	Oct 11, 2009 5:31:1	0	0	0	0
⊟ EchoEJBService						
EchoEJBServicePort	Enabled	Oct 11, 2009 5:31:1	0	0	81	81
⊟ JaxwsWithHandlerChainBeanService						
JaxwsWithHandlerChainBeanPort	Enabled	Oct 11, 2009 5:31:1	0	0	0	0
⊟ CalculatorService						
CalculatorPort	Enabled	Oct 11, 2009 5:31:1	200	0	50	50

Viewing the Web Services in Your Application

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

Using Fusion Middleware Control

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

- View the Web services in the application.
- View the Web service configuration, endpoint status, policy faults, and more.
- View and monitor Web services faults, including Security, Reliable Messaging, MTOM, Management, and Service faults.
- View and monitor Security violations, including authentication, authorization, message integrity, and message confidentiality violations.

- Navigate to pages where you can configure your Web services ports, including enabling and disabling the port, and attaching policies to Web services.

Using WLST

To view the Web services in your application:

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

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

   For example:

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

   moduleName=jaxwsejb,moduleType=web,serviceName=JaxwsWithHandlerChainBeanService
    moduleName=jaxwsejb, moduleType=web, serviceName=WsdlConcreteService
    moduleName=jaxwsejb, moduleType=web, serviceName=EchoEJBService
    moduleName=jaxwsejb, moduleType=web, serviceName=CalculatorService
    moduleName=jaxwsejb, moduleType=web, serviceName=DoclitWrapperWTJService
   ```

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

Viewing the Details for a Web Service Port

Use the procedures described in the following sections to view the details for a Web service port.

Using Fusion Middleware Control

To view the details for a Web service port:

1. Navigate to the Web Services Summary page.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service ports if they are not already displayed.
3. Click the name of the port to navigate to the Web Service Endpoints page.
4. From the Web Service Endpoints page, you can do the following:
   - Click the **Operations** tab to see the list of operations for this port.
   - Click the **Policies** tab to see the policies attached to this port.
   - Click the **Charts** tab to see a graphical display of the faults for this port.
   - Click the **Configuration** tab to see the configuration for this port.

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

Using WLST

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

1. Connect to the running instance of WebLogic Server to which the application is deployed as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
2. Use the `listWebServices` WLST command to display a list of the Web services in your application as described in "Viewing the Web Services in Your Application" on page 6-4.
3. Use the `listWebServicePorts` command to display the port name and endpoint URL for a Web service.

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

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

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

   WsdlConcretePort   http://host.us.oracle.com:7001/jaxwsejb/WsdlAbstract
   ```

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

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

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

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

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

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

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

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

   WsdlConcretePort :
   addressing : oracle/wsaddr_policy , enabled=true
   management : oracle/log_policy , enabled=true
   ```

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

Viewing the Security Violations for a Web Service

Follow the procedure below to view security violations for a Web service.

To view the security violations for a Web service:

1. Navigate to the Web Services Summary page.
2. In the Charts section of the page, select the **Security Violations** tab.

   A graphical representation of the authentication, authorization, confidentiality, and integrity faults for all Web services in the application is displayed in the pie chart.
3. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service 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 **Charts** tab to see a graphical representation of all faults and all security faults.
6. Click the **Policies** tab.

   A list of the policies that are attached to the port is displayed. The status of the policy (whether the policy is enabled or disabled), the number of security faults (authentication, authorization, confidentiality, and integrity), and total policy faults for each policy are displayed.

Viewing Web Service Clients

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

Using Fusion Middleware Control

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

Viewing SOA References

Use the following procedure to view a SOA reference client:

1. From the navigator pane, click the plus sign (+) for SOA deployments, and select the target.
2. Click the Dashboard tab, if it is not already selected.
3. In the Services and References portion of the page, select the SOA reference to view.
4. In the SOA reference page, click the tabs to view the client data.

Viewing ADF DC Web Service Clients

Use the following procedure to view an ADF DC Web service client.

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

   The Application Deployment home page is displayed.
2. Using Fusion Middleware Control, click **Application Deployment**, then click **ADF**.
3. Select the **Administration** tab.
4. Click ADF Connections.
5. In the Web Service Connections portion of the page, click **Configure Web Service**.
6. Select the Web service client endpoint to view.

Viewing WebCenter Portlets

Use the following procedure to view a WebCenter portlet.

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

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

Viewing Asynchronous Web Service Callback Clients

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

1. Navigate to the endpoint for the asynchronous Web service, as described in "Viewing the Details for a Web Service Port" on page 6-5.
2. Click **Callback Client** in the upper right portion of the endpoint page.

Using WLST

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

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

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

For example:

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

/base_domain/soa_server1/soa-infra :
        compositeName=sdo-ejb-cust-nobpel[1.0], moduleType=soa,
serviceRefName=EJBReference
        compositeName=EventMediatorDemo[1.0], moduleType=soa,
serviceRefName=OrderLogger

/base_domain/AdminServer/application1#V2.0 :
        moduleName=test1, moduleType=wsconn, serviceRefName=client
                HelloWorld_pt
                keystore.recipient.alias=A1
                saml.issuer.name=B1
                user.roles.include=C1
                management : oracle/log_policy , enabled=true
                addressing : oracle/wsaddr_policy , enabled=true


/base_domain/soa_server1/ServiceClient :
        moduleName=test1, moduleType=wsconn, serviceRefName=AppModuleService
                AppModuleServiceSoapHttpPort
                security : oracle/wss10_saml_token_client_policy , enabled=true
                management : oracle/log_policy , enabled=true
                addressing : oracle/wsaddr_policy , enabled=true
```

Note that the output displays SOA references (using the `serviceRefName` argument) for the SOA composites `sdo-ejb-cust-nobpel[1.0]` and `EventMediatorDemo[1.0]`. To list the SOA references for a SOA composite, specify the composite name in the command, for example `listWebServiceClients(None,'EventMediatorDemo[1.0]')`.

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

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

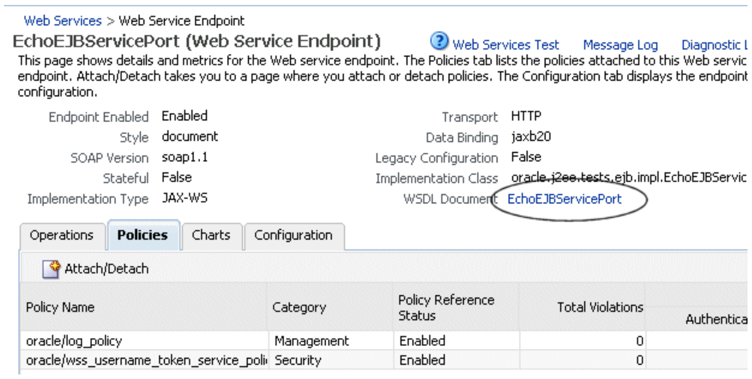
Displaying the Web Service WSDL Document

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

To display the WSDL document for a Web service:

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

Figure 6–3 Web Service Endpoints Page with Web Service WSDL


Web Services > Web Service Endpoint

EchoEJBServicePort (Web Service Endpoint)

This page shows details and metrics for the Web service endpoint. The Policies tab lists the policies attached to this Web servic 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
Style	document	Data Binding	jaxb20
SOAP Version	soap1.1	Legacy Configuration	False
Stateful	False	Implementation Class	oracle.j2ee.tests.ejb.impl.EchoEJBServic
Implementation Type	JAX-WS	WSDL Document	EchoEJBServicePort

Policy Name	Category	Policy Reference Status	Total Violations	Authentica
oracle/log_policy	Management	Enabled	0	
oracle/wss_username_token_service_poli	Security	Enabled	0	

Configuring the Web Service Port

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

Using Fusion Middleware Control

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

1. Navigate to the application's Web Services Summary page, as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service ports if they are not already displayed.
3. Click the name of the port to navigate to the Web Service Endpoints page.
4. Click the **Configuration** tab.
5. Set the configuration attributes and click **Apply.**

   For more information about setting the configuration attributes, see:

   - "Enabling or Disabling a Web Service" on page 6-13
   - "Enabling or Disabling RESTful Web Services" on page 6-14
   - "Enabling or Disabling the Display of the Web Service WSDL Document" on page 6-15
   - "Enabling or Disabling the Exchange of Metadata" on page 6-16
   - "Enabling or Disabling the Web Service Test Endpoint" on page 6-16
   - "Setting the Log Level for Diagnostic Logs" on page 16-3
   - "Validating the Request Message" on page 6-17
   - "Setting the Size of the Request Message" on page 6-18
   - "Configuring Asynchronous Web Services" on page 6-12
6. Restart the application that uses the Web service.

Using WLST

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

1. Connect to the running instance of WebLogic Server to which the application is deployed as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
2. Use the `listWebServices` WLST command to display a list of the Web services in your application as described in "Viewing the Web Services in Your Application" on page 6-4.
3. Use the `listWebServicePorts` command to display the port name and endpoint URL for a Web service.

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

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

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

   WsdlConcretePort   http://host.us.oracle.com:7001/jaxwsejb/WsdlAbstract
   ```

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

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

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

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

   Alternatively, you can set the `detail` argument to `true` in the `listWebServices` command to view the configuration details for the port as shown in "Using WLST" in "Viewing All Current Web Services for a Server" on page 6-2.
5. Use the `setWebServiceConfiguration` command to set or change the port configuration. Specify the properties to be set or changed using the `itemProperties` argument.

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

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

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

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

   For more information about the configurable properties, see:

   - "Enabling or Disabling a Web Service" on page 6-13

- "Enabling or Disabling RESTful Web Services" on page 6-14
- "Enabling or Disabling the Display of the Web Service WSDL Document" on page 6-15
- "Enabling or Disabling the Web Service Test Endpoint" on page 6-16
- "Setting the Size of the Request Message" on page 6-18
- "Setting the Log Level for Diagnostic Logs" on page 16-3

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

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

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

Configuring Asynchronous Web Services

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

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

The following procedure describes how to configure your deployed asynchronous Web services. You can also configure asynchronous *Callback client*, as described in "Configuring Asynchronous Web Service Callback Clients" on page 6-20.

To configure asynchronous Web services:

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

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

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

- Transaction Enabled for Response Queue—Flag that specifies whether transactions are enabled on the response queue.

These flags are configured at design time. For more information, see "Developing Asynchronous Web Services" in *Concepts Guide for Oracle Infrastructure Web Services*.

4. From the Web Service Endpoints page, click the **Configuration** tab.
5. Under the Asynchronous Web Service section of the page, you can set the configuration properties defined in Table 6–1.

Note: The configuration properties defined in Table 6–1 appear and are valid only for asynchronous Web services.

Table 6–1 Configuration Properties for Asynchronous Web Services

Configuration Property	Description
JMS Request Queue Connection Factory Name	Name of the connection factory for the JMS request queue. The default JMS connection factory, weblogic.jms.XAConnectionFactory, provided with the base domain is used by default.
JMS Request Queue Name	Name of the request queue. The following queue is used by default: oracle.j2ee.ws.server.async.DefaultRequestQueue.
JMS Response Queue Connection Factory Name	Name of the connection factory for the JMS response queue. The default JMS connection factory, weblogic.jms.XAConnectionFactory, provided with the base domain is used by default.
JMS Response Queue Name	Name of the request queue. The following queue is used by default: oracle.j2ee.ws.server.async.DefaultResponseQueue.

6. Click **Apply**.
7. Restart the application that uses the Web service.

Enabling or Disabling a Web Service

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

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

Using Fusion Middleware Control

To disable a Web service port:

1. Navigate to the Web Services Summary page.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service ports if they are not already displayed.
3. Click the name of the port to navigate to the Web Service Endpoints page.
4. From the Web Service Endpoints page, click the **Configuration** tab.

5. Select **Disabled** from the Endpoint Enabled control, and click **Apply.**
6. Restart the application that uses the Web service.

Using WLST

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

The procedure for using this command is described in "Using WLST" in "Configuring the Web Service Port" on page 6-10.

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

```
wls:/wls-domain/serverConfig> setWebServiceConfiguration
("/wls-domain/AdminServer/jaxwsejb30ws","jaxwsejb","web","WsdlConcreteService",
"WsdlConcretePort",[("enable","false")])

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

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

Enabling or Disabling RESTful Web Services

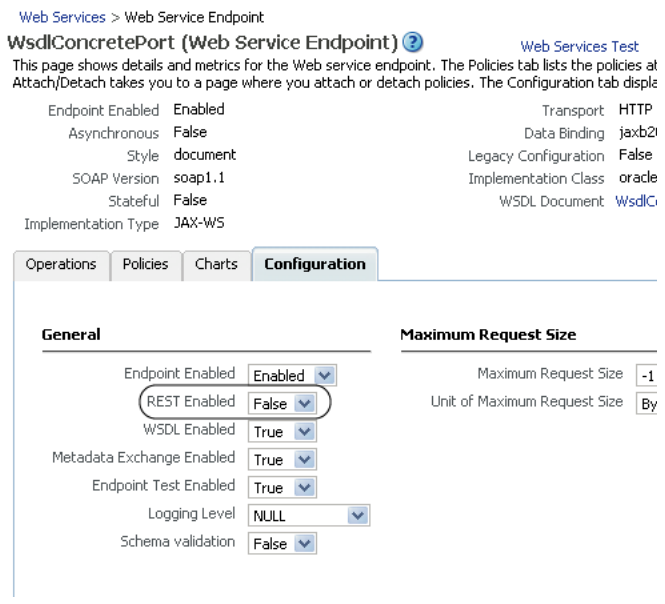
You can enable or disable a Web services port to accept messages in Representational State Transfer (REST) format.

Using Fusion Middleware Control

To enable or disable Web service styles:

1. From the Web Services Summary page, scroll down to the Web Services Details section of the page.
2. Click the **plus sign (+)** of the Web service to display the ports if they are not already displayed.
3. Click the port to display the Web Service Endpoint page.
4. Click the **Configuration** tab.
5. Select **True** from the REST Enabled list to enable REST, or select **False** to disable REST.

Figure 6–4 Enabling and Disabling RESTful Web Services



Web Services > Web Service Endpoint

WsdlConcretePort (Web Service Endpoint)

Web Services Test

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

Endpoint Enabled	Enabled	Transport	HTTP
Asynchronous	False	Data Binding	jaxb2
Style	document	Legacy Configuration	False
SOAP Version	soap1.1	Implementation Class	oracle
Stateful	False	WSDL Document	WsdlC
Implementation Type	JAX-WS		

Operations | Policies | Charts | **Configuration**

General

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

Maximum Request Size

Maximum Request Size	-1
Unit of Maximum Request Size	By

6. Restart the application that uses the Web service.

Using WLST

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

The procedure for using this command is described in "Using WLST" in "Configuring the Web Service Port" on page 6-10.

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

```
wls:/wls-domain/serverConfig> setWebServiceConfiguration
("/wls-domain/AdminServer/jaxwsejb30ws","jaxwsejb","web","WsdlConcreteService",
"WsdlConcretePort",[("enableREST","true")])

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

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

Enabling or Disabling the Display of the Web Service WSDL Document

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

Using Fusion Middleware Control

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

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

3. Click the name of the port to navigate to the Web Service Endpoints page.
4. From the Web Service Endpoints page, click the **Configuration** tab.
5. Select **Enabled** or **Disabled** from the WSDL Enabled control, and click **Apply.**
6. Restart the application that uses the Web service.

Using WLST

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

The procedure for using this command is described in "Using WLST" in "Configuring the Web Service Port" on page 6-10.

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

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

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

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

Enabling or Disabling the Exchange of Metadata

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

To enable or disable the exchange of metadata:

1. Navigate to the Web Services Summary page.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service ports if they are not already displayed.
3. Click the name of the port to navigate to the Web Service Endpoints page.
4. From the Web Service Endpoints page, click the **Configuration** tab.
5. Select **Enabled** or **Disabled** from the Metadata Exchange control, and click **Apply.**
6. Restart the application that uses the Web service.

Enabling or Disabling the Web Service Test Endpoint

The following procedure describes how to enable or disable the Web service test endpoint.

Using Fusion Middleware Control

To enable or disable the Web service test endpoint:

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

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

Using WLST

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

The procedure for using this command is described in "Using WLST" in "Configuring the Web Service Port" on page 6-10.

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

```
wls:/wls-domain/serverConfig> setWebServiceConfiguration
("/wls-domain/AdminServer/jaxwsejb30ws","jaxwsejb","web","WsdlConcreteService",
"WsdlConcretePort",[("enableTestPage","true")])

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

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

Validating the Request Message

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

To enable or disable schema validation:

1. Navigate to the Web Services Summary page.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service ports if they are not already displayed.
3. Click the name of the port to navigate to the Web Service Endpoints page.
4. From the Web Service Endpoints page, click the **Configuration** tab.
5. Select **Enabled** or **Disabled** from the Schema validation control, and click **Apply.**
6. Restart the application that uses the Web service.

Setting the Size of the Request Message

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

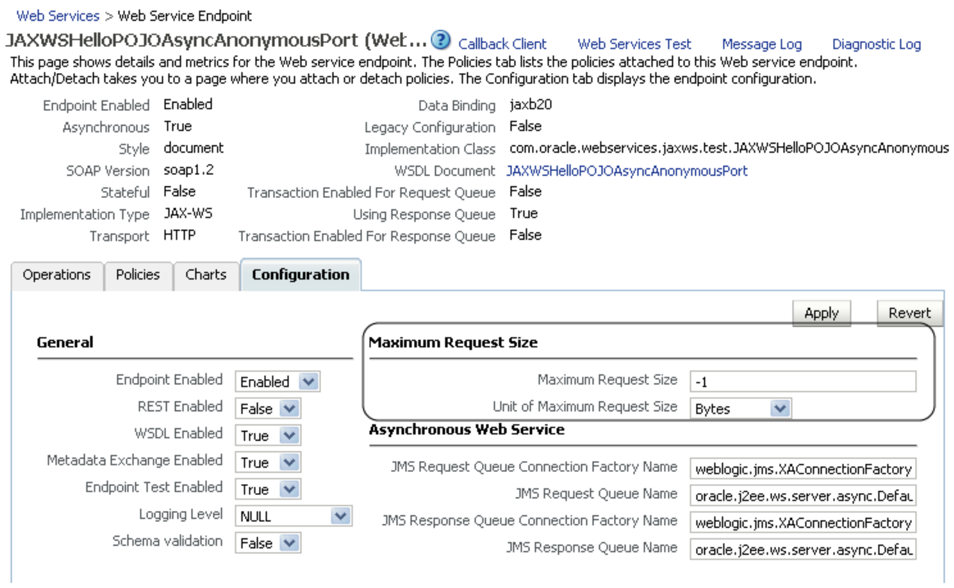
Using Fusion Middleware Control

To set the size of the request message:

1. Navigate to the Web Services Summary page.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service ports if they are not already displayed.
3. Click the name of the port to navigate to the Web Service Endpoints page.
4. Click the **Configuration** tab.
5. Set the Maximum Request Size and the Unit of Maximum Request Size and click **Apply**.

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

   Figure 6–5 Setting Size of Request Message

   Web Services > Web Service Endpoint

   JAXWSHelloPOJOAsyncAnonymousPort (Wet... Callback Client 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	Data Binding	jaxb20
Asynchronous	True	Legacy Configuration	False
Style	document	Implementation Class	com.oracle.webservices.jaxws.test.JAXWSHelloPOJOAsyncAnonymous
SOAP Version	soap1.2	WSDL Document	JAXWSHelloPOJOAsyncAnonymousPort
Stateful	False	Transaction Enabled For Request Queue	False
Implementation Type	JAX-WS	Using Response Queue	True
Transport	HTTP	Transaction Enabled For Response Queue	False

   Operations Policies Charts **Configuration**

   Apply Revert

   General

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

   Maximum Request Size

Maximum Request Size	-1
Unit of Maximum Request Size	Bytes

   Asynchronous Web Service

JMS Request Queue Connection Factory Name	weblogic.jms.XAConnectionFactory
JMS Request Queue Name	oracle.j2ee.ws.server.async.Defau
JMS Response Queue Connection Factory Name	weblogic.jms.XAConnectionFactory
JMS Response Queue Name	oracle.j2ee.ws.server.async.Defau

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

6. Restart the application that uses the Web service.

Using WLST

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

The procedure for using this command is described in "Using WLST" in "Configuring the Web Service Port" on page 6-10.

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

```
wls:/wls-domain/serverConfig> setWebServiceConfiguration
("/wls-domain/AdminServer/jaxwsejb30ws","jaxwsejb","web","WsdlConcreteService",
"WsdlConcretePort",[("maxRequestSize","-1")])
```

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

Enabling and Disabling MTOM

Support for MTOM is provided by attaching the oracle/wsmtom_policy policy to a Web service. You can enable or disable MTOM for a Web service by enabling or disabling this policy. See "Disabling a Policy for a Single Policy Subject" on page 7-17 for more information.

You must restart the application after enabling or disabling MTOM.

Configuring the Web Service Client

For the Web service clients in your application, including SOA references, ADF data control, and asynchronous Web service Callback clients, you can set the configuration properties defined in Table 6–2.

Table 6–2 Configuration Properties for Web Service Clients

Configuration Property	Property Name	Description
Endpoint Address	javax.xml.ws.service.endpoint.address	Port URL to which the client will send the request. **Note**: This property is not available for asynchronous Web service Callback clients.
Maintain Session	javax.xml.ws.session.maintain	Flag that specifies whether the session should be maintained. **Note**: This property is not available for asynchronous Web service Callback clients.
Stop Chunking	oracle.webservices.donotChunk	Flag that specifies whether chunking is enabled for client requests.
Chunking Size (bytes)	oracle.webservices.chunkSize	Size of the request chunk in bytes.
HTTP Read Timeout (ms)	oracle.webservices.httpReadTimeout	Length of the request read timeout in milliseconds.
HTTP Connection Timeout (ms)	oracle.webservices.httpConnTimeout	Length of the request connection timeout in milliseconds.
HTTP User Name	(javax.xml.ws.security.auth.username) oracle.webservices.auth.username	Authenticated HTTP user name.

Table 6–2 (Cont.) Configuration Properties for Web Service Clients

Configuration Property	Property Name	Description
HTTP User Password	(javax.xml.ws.security.auth.password) oracle.webservices.auth.password	Authenticated HTTP user password.
Preemptive	oracle.webservices.preemptiveBasicAuth	Flag that specifies whether security will be sent with the request without being challenged.
Proxy Host	oracle.webservices.proxyHost	URL of proxy to which client will send the request.
Proxy Port	oracle.webservices.proxyPort	Port number of the proxy.
Proxy User Name	oracle.webservices.proxyUsername	Valid user name to access the proxy.
Proxy User Password	oracle.webservices.proxyPassword	Valid password to access the proxy.
Proxy Realm	oracle.webservices.proxyAuthRealm	Realm used by the proxy.
Proxy Authentication Type	oracle.webservices.proxyAuthType	Authentication type used by the proxy.

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

Using Fusion Middleware Control

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

Configuring SOA References

The following procedure describes how to configure a SOA reference.

1. View the SOA reference, as described in "Viewing SOA References" on page 6-7.
2. Click the **Configuration** tab.
3. Set the configuration values as required. Refer to Table 6–2.
4. Click **Apply**.

Configuring ADF DC Web Service Clients

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

1. View the ADF DC Web service client, as described in "Viewing ADF DC Web Service Clients" on page 6-7.
2. Click the **Configuration** tab.
3. Set the configuration values as required. Refer to Table 6–2.
4. Click **Apply**.
5. Restart the application that uses the Web service.

Configuring Asynchronous Web Service Callback Clients

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

To configure an asynchronous Web service Callback client:

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

Using WLST

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

1. Connect to the running instance of WebLogic Server to which the application is deployed as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
2. Use the `listWebServiceClients` WLST command to display a list of the Web service clients in your application as described in "Viewing Web Service Clients" on page 6-7.
3. Use the `listWebServiceClientPorts` command to display the port name and endpoint URL for a Web service client.

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

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

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

   HelloWorld_pt
   ```

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

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

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

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

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

   Alternatively, you can set the `detail` argument to `true` in the `listWebServiceClients` command to view the configuration details for the port as shown in "Using WLST" in "Viewing Web Service Clients" on page 6-7.

5. Do one of the following:
   - Use the `setWebServiceClientStubProperty` command to set or change a single stub property of a Web service client port. Specify the property to be set

or changed using the `propName` and `propValue` arguments. To remove a property, specify a blank value for the `propValue` argument.

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

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

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

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

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

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

For more information about the client properties that you can set, see Table 6–2, " Configuration Properties for Web Service Clients". When specifying these properties, use the format shown in the Property Name column.

You can also set the properties described in "Attaching Client Policies Permitting Overrides" on page 8-13.

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

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

7 Managing Web Service Policies

This chapter includes the following sections:

- Overview of Web Services Policy Management
- Viewing Available Web Services Policies
- Viewing a Web Service Policy
- Searching for Web Service Policies
- Creating Web Service Policies
- Working With Assertions
- Validating Web Services Policies
- Editing Web Service Policies
- Versioning Web Service Policies
- Exporting Web Service Policies
- Deleting Web Service Policies
- Generating Client Policies
- Disabling a Policy for a Single Policy Subject
- Disabling a Web Service Policy for All Subjects
- Analyzing Policy Usage

Overview of Web Services Policy Management

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

Viewing Available Web Services Policies

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

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

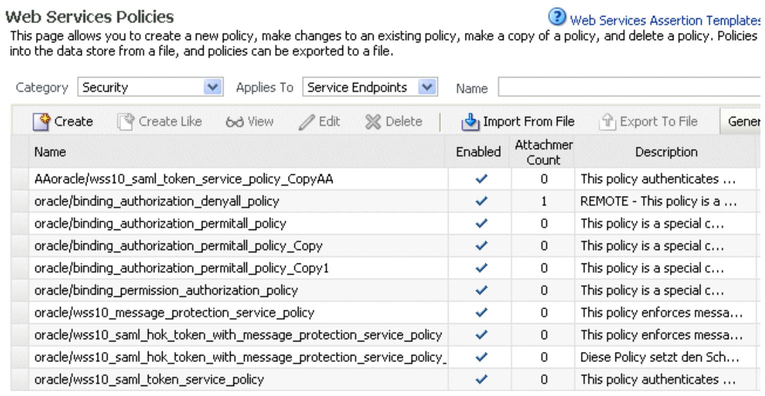
Navigating to the Web Services Policies Page in Fusion Middleware Control

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

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

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

Figure 7–1 Web Services Policy Page



Web Services Policies

Web Services Assertion Template

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

Category: Security | Applies To: Service Endpoints | Name:

Name	Enabled	Attachmer Count	Description
AAoracle/wss10_saml_token_service_policy_CopyAA	✓	0	This policy authenticates ...
oracle/binding_authorization_denyall_policy	✓	1	REMOTE - This policy is a ...
oracle/binding_authorization_permitall_policy	✓	0	This policy is a special c...
oracle/binding_authorization_permitall_policy_Copy	✓	0	This policy is a special c...
oracle/binding_authorization_permitall_policy_Copy1	✓	0	This policy is a special c...
oracle/binding_permission_authorization_policy	✓	0	This policy is a special c...
oracle/wss10_message_protection_service_policy	✓	0	This policy enforces messa...
oracle/wss10_saml_hok_token_with_message_protection_service_policy	✓	0	This policy enforces messa...
oracle/wss10_saml_hok_token_with_message_protection_service_policy_	✓	0	Diese Policy setzt den Sch...
oracle/wss10_saml_token_service_policy	✓	0	This policy authenticates ...

Displaying a List of the Available Policies Using WLST

To display a list of the available policies using WLST:

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

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

   For example:

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

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

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

   For example:

```
wls:/base_domain/domainRuntime>
listAvailableWebServicePolicies("security","server")
List of available OWSM policy - total : 27
security : oracle/wss10_saml_token_with_message_integrity_service_policy
security : oracle/wss11_username_token_with_message_protection_service_policy
security : oracle/wss11_x509_token_with_message_protection_service_policy
security : oracle/wss_username_token_service_policy
security : oracle/wss_saml_token_over_ssl_service_policy
security : oracle/wss10_x509_token_with_message_protection_service_policy
security : oracle/wss10_username_token_with_message_protection_ski_basic256_
service_policy
```

Viewing a Web Service Policy

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

To view a Web service policy

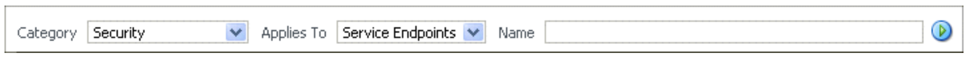
1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select a policy from the Policies table and click **View.**
3. When you are done viewing the policy, click **Return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

Searching for Web Service Policies

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

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

Figure 7–2 Search Filter Criteria


Category Security Applies To Service Endpoints Name

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

Table 7–1 Search Filter Criteria

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

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

Creating Web Service Policies

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

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

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

Creating a New Web Service Policy

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

To create a new Web service policy

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.

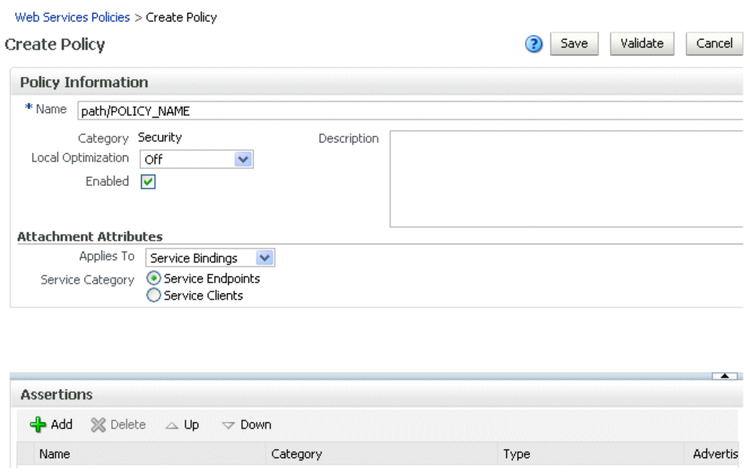
2. From the Category list, select the category to which this policy will belong and click **Create**.

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

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

   Oracle recommends that you follow the policy naming conventions described in "Recommended Naming Conventions for Policies" on page 3-9.

Figure 7–3 Create Policy Page



Web Services Policies > Create Policy

Create Policy

Policy Information

* Name: path/POLICY_NAME

Category: Security

Local Optimization: Off

Enabled: ✓

Description:

Attachment Attributes

Applies To: Service Bindings

Service Category: (●) Service Endpoints ( ) Service Clients

Assertions

Name	Category	Type	Advertis

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

4. Set the Local Optimization control. See "Configuring Local Optimization" on page 10-66 for a description of the Local Optimization control.

5. By default, the policy is enabled. If you want to disable the policy, clear the **Enabled** box. A policy that is not enabled is not enforced at runtime.

6. Specify the type of policy subjects the policy can be attached to by selecting from the Applies To list. If you select Service Bindings, then specify whether the policy can be attached to Web service endpoints, Web service clients, or to both.

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

7. To add a single assertion:

   a. n the Assertion Information section, click **Add**.

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

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

   c. Click **OK**.

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

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

Creating a Web Service Policy from an Existing Policy

You can take a Web service policy and use it as a base for creating another policy. By default, Oracle Fusion Middleware 11*g* Release 1 (11.1.1) comes with predefined policies. You can create a copy of one of the predefined policies or you can create a copy of a policy that you have created. Once the policy is created, you can treat it like any other policy, adding or deleting assertions, and modifying existing assertions.

To make a copy of a Web service policy

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

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

   It is recommended that you change the name of this new policy to be more meaningful in your environment.
4. Modify the policy as required, including the assertions.

5. Click **Validate** to verify that the policy does not contain errors. For more information on policy validation, see "Validating Web Services Policies" on page 7-9.
6. Click **Save.**

Importing Web Service Policies

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

Note: The policy name you import must not already exist in the Policy Store.

Be aware that "policy name" and "file name" are different. The policy name is specified by the name attribute of the policy content; the file name is the name of the policy file. You might find it convenient for the two names to match, but it is not required.

You cannot prefix the name of a policy with oracle_. Otherwise, you will receive exceptions when you try to use the policy.

To import a Web service policy

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

Creating Custom Policies

For information about creating custom Web service policies, see "Creating Custom Assertions" on page 14-1.

Working With Assertions

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

The following sections provide more information about working with assertions:

- "Naming Conventions for Assertion Templates" on page 7-8
- "Viewing an Assertion Template" on page 7-8
- "Adding Assertions to a Policy" on page 7-8
- "Adding an OR Group to a Policy" on page 7-8
- "Configuring Assertions" on page 7-9

Naming Conventions for Assertion Templates

The same naming conventions used to name predefined policies are used to name the assertion templates. Assertion templates begin with the directory name *oracle/* and are identified with the suffix *_template* at the end; for example, *oracle/wss10_message_protection_service_template*. For more information on naming conventions for predefined policies, see "Recommended Naming Conventions for Policies" on page 3-9.

Viewing an Assertion Template

To view the assertion templates, from the Web Services Policies page navigate to the Web Services Assertion Templates page. By default, you will see all of the assertion templates in the list.

Select the template you want to view from the list and click **View**.

Adding Assertions to a Policy

You can add assertions from the Create Policy page, the Copy Policy page, or the Edit Policy Detail page.

Each policy can contain only one assertion for each of the following categories: MTOM Attachments and Reliable Messaging. The policy can contain any number of assertions belonging to the Security category; however, the combination of assertions must be valid. For more information on valid assertions, see "Validating Web Services Policies" on page 7-9.

To add an assertion to a policy:

1. Navigate to the Create Policy page, the Create Like page, or the Edit Policy Detail page.
2. In the Assertion List section, click **Add**.
3. In the Add an Assertion dialog, enter the name of your assertion, and select an assertion from the Assertion Template list.
4. Click **OK**.

Adding an OR Group to a Policy

You can create an OR group, consisting of one or more assertions, enabling a single policy to accept multiple types of security tokens. A client can enforce *any one* of the policies that are defined in the OR group. For more information, see "Defining Multiple Policy Alternatives (OR Groups)" on page 3-8.

You can add only one OR group to a policy. Once you have generated an OR Group, the Add OR Group button is greyed out.

You can add an OR group from the Create Policy page, the Copy Policy page, or the Edit Policy Detail page.

To add an OR group to a policy:

1. Navigate to the Create Policy page, the Create Like page, or the Edit Policy Detail page.
2. In the Assertion List section, click **Add OR Group**.
3. In the Add OR Group dialog, enter the name of the first assertion in the group, and select an assertion template from the Assertion Template list.

4. Click **OK**.

   The assertion is added under the OR Group.

5. To add additional assertions to the OR group:

   a. Ensure that an assertion within the OR group is currently selected.

   b. Click **Add**.

   c. In the Add Assertion dialog, enter the name of the assertion in the group, and select an assertion template from the Assertion Template list.

   d. Click **OK**.

6. To configure the assertions, see "Configuring Assertions" on page 7-9.

   The policy attribute values for attachTo and category limit the assertions that are valid within the current policy. All assertions within an OR group must be compatible with the attachTo and category attribute values in order to be considered.

   o

7. When you have finished adding assertions to the OR group, select the assertions and use the **Up** and **Down** controls to order them as needed. Assertions are considered for invocation in the order that they appear on the list.

8. To delete an assertion from the OR group, select the assertion and click **Delete**. To delete the entire OR group, select the OR group and click **Delete**.

Configuring Assertions

Once an assertion has been added to a policy, you can configure the assertion attributes. You can configure assertions from the Create Policy page, the Create Like page, or the Edit Policy Detail page.

To configure an assertion:

1. Navigate to the Create Policy page, the Create Like page, or the Edit Policy Detail page.

2. Select the assertion in the assertion table.

3. In the Assertion Details section of the page, click one of the tabs, **Settings** or **Configurations**.

4. Edit the Settings and Configuration attributes, and click **Save.**

See Appendix C, "Predefined Assertion Templates" for more information about the Settings and Configuration assertion attributes.

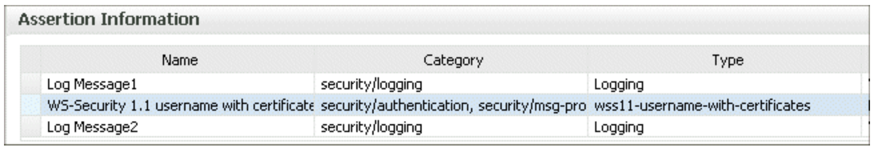
Validating Web Services Policies

There are restrictions on the type and number of policy assertions that are permitted in a Web service policy. When you validate a policy, Enterprise Manager checks to see if the policy is consistent with these restrictions. A policy can contain only assertions that belong to a single category. Therefore, you cannot combine a Security assertion with an MTOM assertion in the same policy. The policy type is determined by the category of the assertion. Therefore, a policy containing a security assertion is a security policy, a policy containing a management assertion is a management policy, and so on. Security assertions are further categorized into subcategories: authentication, logging, message protection (msg-protection), and authorization.

There are restrictions on the number and type of assertions you can have in a policy. The restrictions are as follows:

- MTOM and Reliable Messaging policies can contain only one assertion.
- A security policy can contain multiple security assertions; however, there can be only one assertion of each subcategory in a policy.
- Some assertions contain both authentication and message protection. For example, if you view the *oracle/wss11_username_token_with_message_protection_service_policy*, you will see that the second assertion falls into two categories: security/authentication and security/msg-protection. See Figure 7–4.

Figure 7–4 Assertion Belonging to Two Categories



Assertion Information

Name	Category	Type
Log Message1	security/logging	Logging
WS-Security 1.1 username with certificate	security/authentication, security/msg-pro	wss11-username-with-certificates
Log Message2	security/logging	Logging

- A security policy can contain any number of security_log_template assertions. For example, if you view any of the predefined security policies, you will see two logging assertions included.

Oracle recommends that you create one policy for authentication and message protection, and a second policy for authorization. If you create a policy that contains both an authentication and an authorization assertion, then the authentication assertion must precede the authorization assertion.

When you validate your policies, the validation process checks to see that your policies meet these requirements. If the validation fails during policy creation, the policy is created but is marked as disabled.

Validating a Policy

Policies can be validated from the Create Policy and Edit Policy pages.

To validate a policy:

1. From the Create Policy or Edit Policy page, make any changes to your policy.
2. Click Validate.

   If successful, the *Validation successful* message appears.

   If not successful, the resulting error message describes the problem.

Editing Web Service Policies

You can make changes to the policies you create or to the predefined policies that come with the product. However, Oracle recommends that you do not change the predefined policies so that you will always have a known set of valid policies to work with.

The changes take effect at the next polling interval for policy changes. If you are using a database-based metadata repository, each time you save a change to your policy, a new version is created, and the older versions are retained.

To edit Web service policies:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select a policy from the Policies table and click **Edit**.
3. On the Edit Policy page, make the changes to the policy.
4. Click **Save**.

Versioning Web Service Policies

Whenever a change to a policy is saved, this results in a new version of the policy being automatically created and the version number being incremented. The Policy Manager maintains the history of these changes, and you can go back to an earlier version.

For example, you might find it useful to create two different versions of a policy, perhaps one with logging and one without, and alternate between them. As another example, you might have an occasional need to use a policy such as *oracle/binding_authorization_denyall_policy* policy with selected roles to temporarily lock down access to a Web service.

By using the versioning feature, you can reuse multiple versions of a policy without having to recreate them every time you need them.

The following sections describe versioning in more detail:

- "Viewing the Version History of Web Services Policies" on page 7-11
- "About the Restore and Activate Policy Options" on page 7-12
- "Creating a New Version of a Web Service Policy" on page 7-13
- "Restoring an Earlier Version of a Web Service Policy" on page 7-13
- "Deleting Versions of a Web Service Policy" on page 7-14

Note: The versioning feature described in this section requires that you use a database-based Metadata Service (MDS). If you are not using a database-based MDS, versioning information is not maintained or displayed.

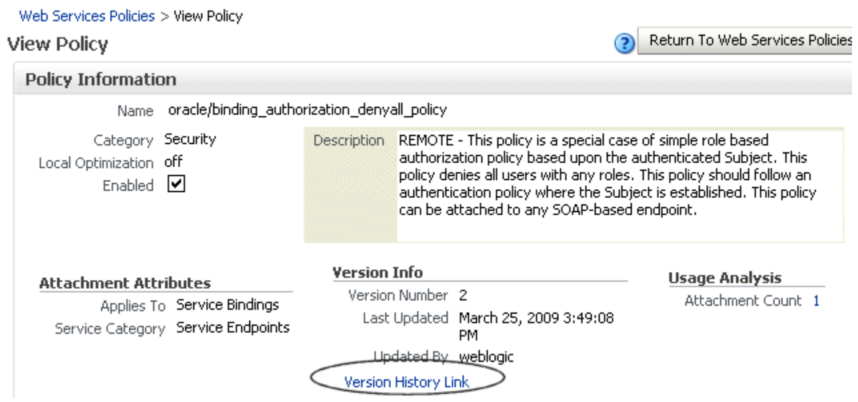
Viewing the Version History of Web Services Policies

To view the Web services policy version history:

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

   In the Policy Information section, you see the version information, including the Version Number of the active version and the date that the policy was last updated.
3. In the View Policy page, click **Version History Link** (Figure 7–5) to go to the View Policy Version History page.

Figure 7–5 Version History Link on the Edit Policy Page


Web Services Policies > View Policy
View Policy
Return To Web Services Policies
Policy Information

Name oracle/binding_authorization_denyall_policy

Category Security
Local Optimization off
Enabled ☑

Description REMOTE - This policy is a special case of simple role based authorization policy based upon the authenticated Subject. This policy denies all users with any roles. This policy should follow an authentication policy where the Subject is established. This policy can be attached to any SOAP-based endpoint.

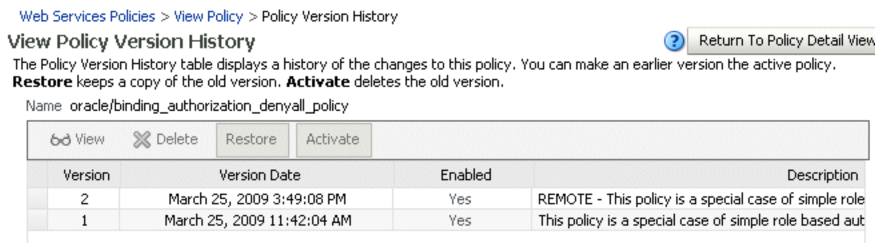
Attachment Attributes
Applies To Service Bindings
Service Category Service Endpoints

Version Info
Version Number 2
Last Updated March 25, 2009 3:49:08 PM
Updated By weblogic
Version History Link

Usage Analysis
Attachment Count 1

4. The policies appear in order in the Policy Version History table with the active policy shown first (Figure 7–6). The active policy has the highest version number, and is the only policy that can be attached to a subject. However, you can make an earlier version of a policy the active policy.

Figure 7–6 View Policy Version History Page


Web Services Policies > View Policy > Policy Version History
View Policy Version History
Return To Policy Detail View

The Policy Version History table displays a history of the changes to this policy. You can make an earlier version the active policy. **Restore** keeps a copy of the old version. **Activate** deletes the old version.

Name oracle/binding_authorization_denyall_policy

View
Delete
Restore
Activate

Version	Version Date	Enabled	Description
2	March 25, 2009 3:49:08 PM	Yes	REMOTE - This policy is a special case of simple role
1	March 25, 2009 11:42:04 AM	Yes	This policy is a special case of simple role based aut

About the Restore and Activate Policy Options

You can make an earlier version active by selecting a policy from the Policy Version History table (Figure 7–6), and clicking either the Restore or Activate Policy buttons. In both instances, the selected policy is made the current, active policy, and the policy version number is incremented. The following describes the difference between the Restore and Activate Policy options:

- Clicking **Restore**, the earlier version of the policy is retained. You can make the earlier version the active version without deleting it. Use Restore if you are modifying your policy and want to keep earlier versions of the policy.
- Clicking **Activate Policy**, the selected policy is now the current active policy. The earlier version of the policy is deleted, and the current version is incremented by 1. For example, assume that you have version 1 and version 3 of the policy. You select version 1 and click **Activate Policy**. The policy is activated as version 4, and version 1 is deleted.

  The Activate Policy option can be used in situations where you need to switch between different versions, but you do not want to keep adding policy versions. For example, you may use one version of the policy during business hours and another version during non-business hours. You want to switch between the versions, but you do not want to accumulate multiple versions of the same policy. Therefore, you use Activate Policy to delete the earlier version.

You can also delete any version of the policy, except the active policy, from the Policy Version History table by selecting the policy and clicking **Delete.** You cannot edit the policy from the Policy Version History page. You must edit a policy from the Web Services Management page.

Creating a New Version of a Web Service Policy

You create a new version of an existing Web service policy by making any desired changes and saving the policy.

Note: Save does an implicit validation. If the validation fails, the policy is persisted, but the status is set to **Disabled**.

To create a new version of a Web service policy:

1. From the Edit Policy page, make a change to your policy.
2. Click **Save.**

In the Policy Information section of the page, the version number for the policy is incremented by 1.

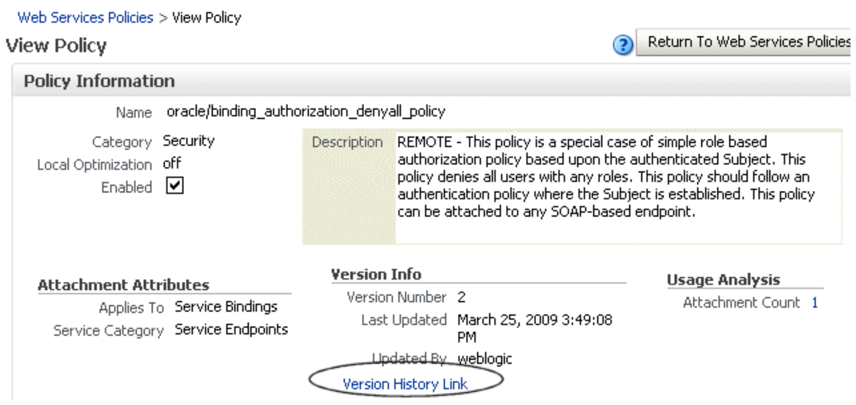
Restoring an Earlier Version of a Web Service Policy

Follow the procedure below to return to an earlier version of a policy.

To restore an earlier version of a Web service policy

1. From the View Policy page, click **Version History Link**, as shown in Figure 7–7.

Figure 7–7 Version History Link on Edit Policy Page


Web Services Policies > View Policy
View Policy
Return To Web Services Policies
Policy Information

Name oracle/binding_authorization_denyall_policy

Category	Security
Local Optimization	off
Enabled	✓

Description REMOTE - This policy is a special case of simple role based authorization policy based upon the authenticated Subject. This policy denies all users with any roles. This policy should follow an authentication policy where the Subject is established. This policy can be attached to any SOAP-based endpoint.

Attachment Attributes

Applies To	Service Bindings
Service Category	Service Endpoints

Version Info

Version Number	2
Last Updated	March 25, 2009 3:49:08 PM
Updated By	weblogic

Version History Link

Usage Analysis

Attachment Count	1

2. In the Policy History table, select a policy and click **Restore** or click **Activate Policy** .

Note: *Restore* saves the earlier version of the policy, and *Activate Policy* deletes the earlier version.

If you click **Restore**, the selected policy is now the current active policy. The earlier version of the policy is retained, and the current version is incremented by 1.

If you click **Activate Policy**, the selected policy is now the current active policy. The earlier version of the policy is deleted, and the current version is incremented by 1.

Deleting Versions of a Web Service Policy

Follow the procedure below to permanently remove earlier versions of a policy. You can delete all versions except the active policy version. To delete all versions of the policy, including the active version, see "Deleting Web Service Policies" on page 7-14.

To delete a Web service policy version

1. From the Copy Policy page or the Edit Policy Detail page, click **Version History Link**.
2. In the Policy History table, select the policy want to remove, and click **Delete**.
3. A dialog box appears with a message asking you to confirm the deletion. Click **OK**.

The selected policy is deleted from the Metadata Service and the Policy History table.

Exporting Web Service Policies

You might want to export a policy to copy it from a development environment to a production environment, or to simply view the policy in another tool or application. Follow the procedure in this section to export a policy from the policy store. Once the policy is exported, you can import it to another policy store, attach it to Web services, make changes to it, and so forth.

To export a Web service policy

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. Select the policy that you want to export from the list.
3. From the Web Services Policies page, click **Export to File.**
4. Save the policy in the filename of your choice. (Use only ASCII characters in the filename.)

Note: You cannot prefix the name of a policy with oracle_. When you export a predefined policy file, the file is renamed from oracle/<*policyname*> to oracle_<*policyname*>. You should change this name. Otherwise, you will receive exceptions when trying to use the policy.

Deleting Web Service Policies

Before you delete a policy, Oracle recommends that you verify that the policy is not attached to any policy subjects. You can see the policy subjects that are attached to a policy by doing a policy dependency analysis. See "Analyzing Policy Usage" on page 7-19 for more information. If you try to delete a policy that is attached to a subject, you will receive a warning. You will not be prevented from deleting an attached policy. However, the Web service request will fail the next time the subject to which the policy is attached is invoked.

When you delete a policy, the active policy and all previous versions of the policy are deleted. To retain the active policy version and delete only the previous versions of the policy, see "Versioning Web Service Policies" on page 7-11.

To delete a Web service policy:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select a policy from the Policies table and click **Delete**.
3. A dialog box appears asking you to confirm the deletion. Click **OK**.

Generating Client Policies

Once you have created the service policy, you can use the Web service WSDL to generate an equivalent client policy with the parameters required to call that service.

You must use the Oracle WSDL instead of the standard WSDL to generate the client policy. The URL for the Web service must be appended with *?orawsdl*, instead of *?wsdl.* Generating the policy increases the likelihood that the client policy will work with the service policy.

Once a policy is generated, you can edit the policy. The policy is populated with the client assertion that is the matching pair to the service assertion. For example, if the service policy contained the assertion, wss_http_token_*service*_template, then the generated client policy is populated with its counterpart, wss_http_token_*client*_template.

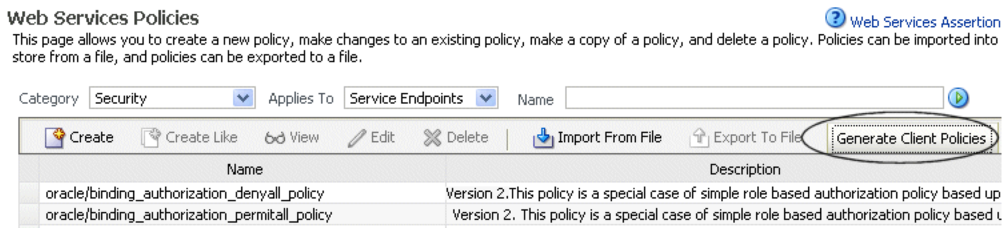
However, the client security policies that are generated will not contain any configuration information. Therefore, once the policies are generated, use the client assertion template and import the configuration information into your client policy. In the example, you would import configuration information from the client assertion template, *wss_http_token_client_template*. After you have made the desired changes to the policy, you must save the policy. Once a policy is saved, you can access it from the Web Services Management page.

You can also delete any generated policies that you do not need. For example, you may want to delete duplicates of already existing MTOM or Reliable Messaging policies.

To generate a Web service client policy

1. Determine the WSDL for the Web service for which you want to generate a Web service client policy.
2. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
3. From the Web Services Policies page, click **Generate Client Policies**, as shown in Figure 7–8.

Figure 7–8 Generate Client Policies on the Web Services Policies Page


Web Services Policies

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

Category: Security; Applies To: Service Endpoints; Name:

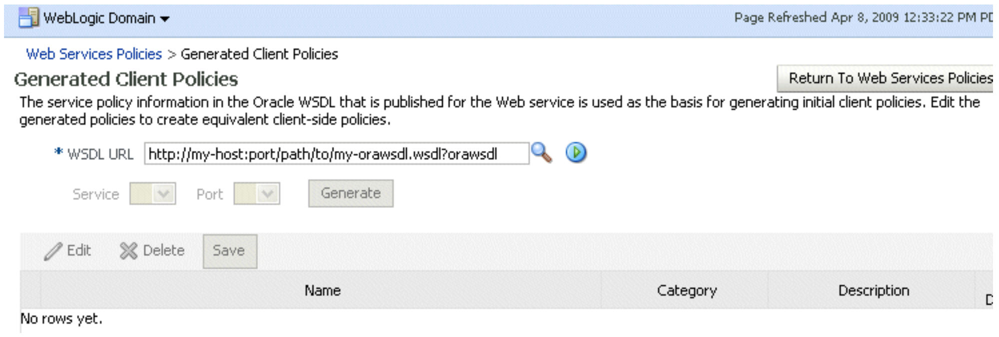
Name	Description
oracle/binding_authorization_denyall_policy	Version 2.This policy is a special case of simple role based authorization policy based up
oracle/binding_authorization_permitall_policy	Version 2. This policy is a special case of simple role based authorization policy based u

4. In the Generated Client Policies page, enter the URL to the Web service WSDL using the following format: *Web_service_endpoint*?orawsdl, and click the control to access the Web service and ports, as shown in Figure 7–9.

> **Note:** You must use *?orawsdl*, instead of *?wsdl*, to get the WSDL that is used to generate the corresponding client policy. Prepend *ora* to *wsdl* to accomplish this.

The *Web_service_endpoint* is the URL to the Web service. The service policy information in the Oracle WSDL published for the Web service is used as the basis for generating the initial client policies.

Figure 7–9 Getting the Web Service and Ports


WebLogic Domain

Page Refreshed Apr 8, 2009 12:33:22 PM PD

Web Services Policies > Generated Client Policies

Generated Client Policies

The service policy information in the Oracle WSDL that is published for the Web service is used as the basis for generating initial client policies. Edit the generated policies to create equivalent client-side policies.

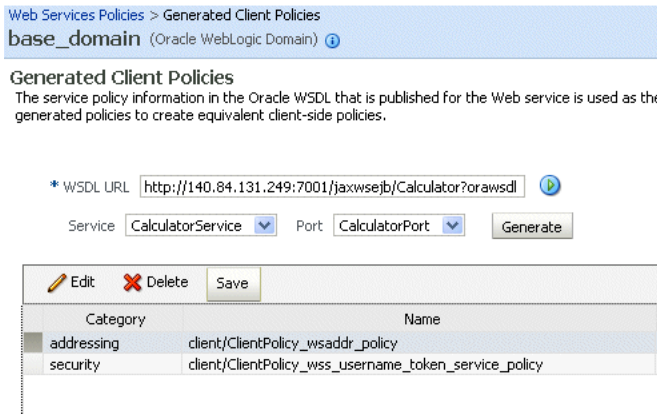
* WSDL URL: http://my-host:port/path/to/my-orawsdl.wsdl?orawsdl

Name	Category	Description

No rows yet.

5. In the Generated Client Policies page (Figure 7–10), click **Generate** to generate the client policies, as shown in Figure 7–10.

Figure 7–10 Generated Client Policies Page


Web Services Policies > Generated Client Policies

base_domain (Oracle WebLogic Domain)

Generated Client Policies

The service policy information in the Oracle WSDL that is published for the Web service is used as the generated policies to create equivalent client-side policies.

* WSDL URL: http://140.84.131.249:7001/jaxwsejb/Calculator?orawsdl

Service: CalculatorService; Port: CalculatorPort

Category	Name
addressing	client/ClientPolicy_wsaddr_policy
security	client/ClientPolicy_wss_username_token_service_policy

6. Select a generated policy from the table and click **Edit**.
7. In the Edit Policy page, edit the policy as necessary.
8. Click **Validate** to validate your changes.
9. Click **Save** to save the changes to your policy.
10. You are returned to the Generated Client Policies page. Edit the other policies as needed.

Once the policy is saved, you can navigate to the Web Services Management page and find the policy in the Policies table.

Disabling a Policy for a Single Policy Subject

When a policy is attached to a Web service, it is enabled by default. You may temporarily disable a policy for a single endpoint without disassociating it from the Web service. When the policy is disabled for an endpoint, it is not enforced for that endpoint.

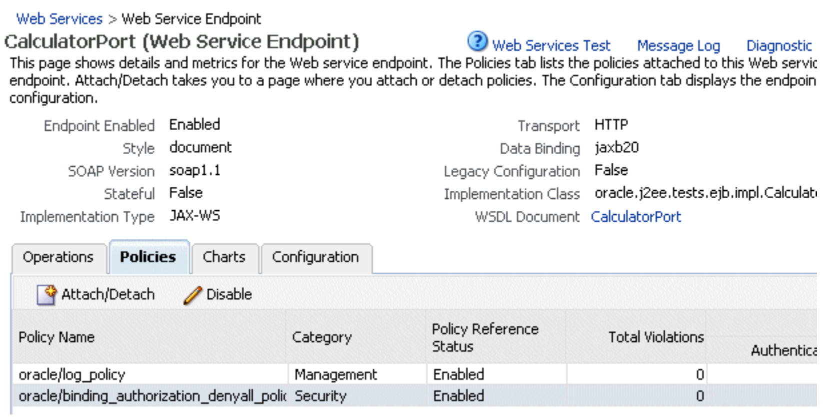
Using Fusion Middleware Control

Policies must be individually enabled or disabled for the endpoint; you cannot enable or disable multiple policies at the same time.

To disable a policy attachment:

1. From the Web Service Endpoints page, click the **Policies** tab.
2. Select the policy you want to disable.
3. Select Disable and confirm your selection. (See Figure 7–11.)

Figure 7–11 Disabling a Policy Attachment



Web Services > Web Service Endpoint

CalculatorPort (Web Service Endpoint)

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

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

Policy Name	Category	Policy Reference Status	Total Violations	Authentica
oracle/log_policy	Management	Enabled	0	
oracle/binding_authorization_denyall_polic	Security	Enabled	0	

Using WLST

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

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

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

For example, to see a list of the policies attached to the `WsdlConcretePort`, use the following command:

```
wls:/base_domain/domainRuntime> listWebServicePolicies('/base_domain/soa_
server1/jaxwsejb30ws',
'jaxwsejb','web','WsdlConcreteService','WsdlConcretePort')

WsdlConcretePort :
security : oracle/binding_authorization_denyall_policy , enabled=true
security : oracle/wss_username_token_service_policy , enabled=true
```

3. Disable a single policy using the `enableWebServicePolicy` command and setting the `enable` argument to `false`.

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

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

```
wls:/base_domain/domainRuntime> enableWebServicePolicy('/base_domain/soa_
server1/jaxwsejb30ws',
'jaxwsejb','web','WsdlConcreteService','WsdlConcretePort','oracle/binding_
authorization_denyall_policy',false)
```

4. Disable multiple policies attached to a port using the `enableWebServicePolicies` command and setting the `enable` argument to `false`.

```
enableWebServicePolicies(application, moduleOrCompName, moduleType,
serviceName, subjectName,
policyURIs,[enable],[subjectType=None] ))
```

For example:

```
wls:/base_domain/domainRuntime> enableWebServicePolicies('/base_domain/soa_
server1/jaxwsejb30ws',
'jaxwsejb','web','WsdlConcreteService','WsdlConcretePort',
['oracle/binding_authorization_denyall_policy',oracle/wss_username_token_
service_policy],false)
```

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

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

Disabling a Web Service Policy for All Subjects

When a policy is created, it is enabled by default unless it has validation errors. A policy can be globally disabled from the Edit Policy page. You can disable the policy from one central location, and it will be disabled for any policy subject to which it is attached.

When you disable a policy from the Edit Policy page, the policy continues to be attached to the policy subjects, but the policy is not enforced. You may want to temporarily disable a policy if you discover that there is a problem with the policy that is causing all requests to a Web service to fail. Once the problem is corrected, you can globally enable the policy.

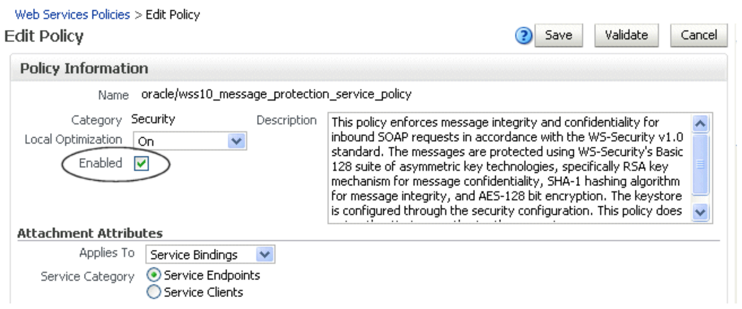
Before disabling a policy, you may want to click **Usage Analysis Link** (see "Analyzing Policy Usage" on page 7-19) to see which policy subjects the policy is attached to. The change to the policy takes effect at the next polling interval for policy changes.

You may also selectively disable a policy for a specific policy subject rather than for all policy subjects. See "Disabling a Policy for a Single Policy Subject" on page 7-17 for more information.

To disable a Web service policy for all policy subjects

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. Select a policy from the Policies table and click **Edit**.
3. In the Policy Information section of the Edit Policy page, clear the **Enabled** box (Figure 7–12).

Figure 7–12 Enabled Box on the Edit Policy Page



Name oracle/wss10_message_protection_service_policy

Category Security

Local Optimization On

Enabled ✓

Description This policy enforces message integrity and confidentiality for inbound SOAP requests in accordance with the WS-Security v1.0 standard. The messages are protected using WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. The keystore is configured through the security configuration. This policy does

Applies To Service Bindings

Service Category [✓ Service Endpoints] Service Clients

4. Click **Save**.

Analyzing Policy Usage

Note: The policy usage feature described in this section requires that you use a database-based Metadata Service (MDS). If you are not using a database-based MDS, policy usage information is not maintained or displayed.

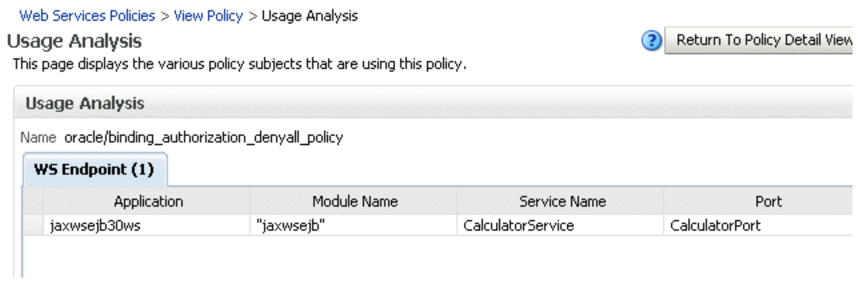
Policies are created at and managed from the domain level. The central management of policies gives you the ability to reuse policies and attach them to multiple policy subjects. Any change to a policy (for example, editing a policy or deleting a policy) affects all policy subjects to which the policy is attached. Therefore, before making any changes to your policies, Oracle recommends you do a usage analysis to see which subjects are using a particular policy.

Note: The usage analysis simply identifies which policy subjects will be affected; it does not define the effect of the change. You need to evaluate the change on each of the policy subjects and determine if you should proceed.

To perform a usage analysis:

1. Navigate to the Web Services Policies page.
2. The Subject Count column of the Policies table shows the number of subjects to which a policy is attached.
3. Select the policy from the Policies table and click **View.**
4. In the Policy Information region of the page, click the **Attachment Count Number** to display the Usage Analysis page.

Figure 7–13 Usage Analysis for a Policy



Web Services Policies > View Policy > Usage Analysis

Usage Analysis

This page displays the various policy subjects that are using this policy.

Usage Analysis

Name oracle/binding_authorization_denyall_policy

WS Endpoint (1)

Application	Module Name	Service Name	Port
jaxwsejb30ws	"jaxwsejb"	CalculatorService	CalculatorPort

The Usage Analysis table shows the different policy subjects to which this policy is attached. Subjects are organized into the following categories: Service Reference, SOA Component, SOA Reference, SOA Service, WS Endpoint, Async Callback Client (asynchronous Callback clients), and WS Connection.

8

Attaching Policies to Web Services

This chapter includes the following sections:

- Viewing the Policies That are Attached to a Web Service
- Attaching a Policy to a Single Subject
- Attaching a Policy to Multiple Subjects (Bulk Attachment)
- Validating Policy Subjects
- Attaching Policies to Web Service Clients
- Attaching Client Policies Permitting Overrides
- Attaching Web Service Policies Permitting Overrides
- Configuring User-Defined Client- or Server-Side Override Properties
- Overriding Configuration Properties When Attaching a Policy Using WLST

Viewing the Policies That are Attached to a Web Service

The following sections describe how to view the policies that are attached to a Web service using Fusion Middleware Control and the WebLogic Scripting Tool (WLST).

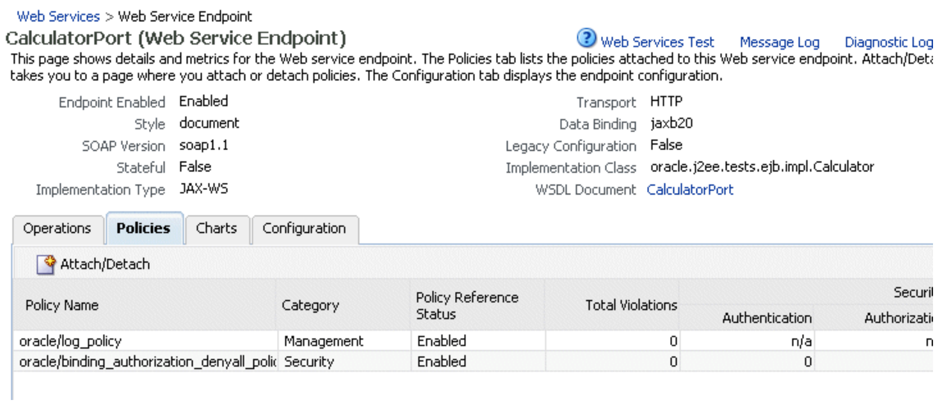
Using Fusion Middleware Control

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

1. Navigate to the home page for the Web service, as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service ports if they are not already displayed.
3. Click the name of a port to navigate to the Web Service Endpoints page for a particular Web service.
4. Click the **Policies** tab.

   A list of the policies that are attached to the port is displayed, as shown in Figure 8–1.

Figure 8–1 Policies Attached to a Web Service



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/Det
takes you to a page where you attach or detach policies. The Configuration tab displays the endpoint configuration.

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

Policy Name	Category	Policy Reference Status	Total Violations	Securi	
				Authentication	Authorizati
oracle/log_policy	Management	Enabled	0	n/a	n
oracle/binding_authorization_denyall_poli	Security	Enabled	0	0	

Using WLST

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

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

2. Use the `listWebServices` WLST command to display a list of the Web services in your application as described in "Viewing the Web Services in Your Application" on page 6-4.

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

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

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

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

   WsdlConcretePort    http://host.us.oracle.com:7001/jaxwsejb/WsdlAbstract
   ```

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

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

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

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

   WsdlConcretePort :
   addressing : oracle/wsaddr_policy , enabled=true
   management : oracle/log_policy , enabled=true
   ```

Attaching a Policy to a Single Subject

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

The order in which policies are attached to a subject or appear in the list of attached polices does not determine the order in which policies are executed. As a message is passed between the client and the Web service, the order of the interceptors in the policy interceptor chain determines the order in which the policies are executed.

See "How Policies are Executed" on page 3-6 for more information.

Note: Policy attachment is not synchronized automatically for ADF and WebCenter Web services in a cluster. When using ADF and WebCenter Web services in a cluster, you must attach and/or detach policies to each instance of the cluster. This issue does not apply to SOA composite applications.

Attaching a Policy to a Web Service Using Fusion Middleware Control

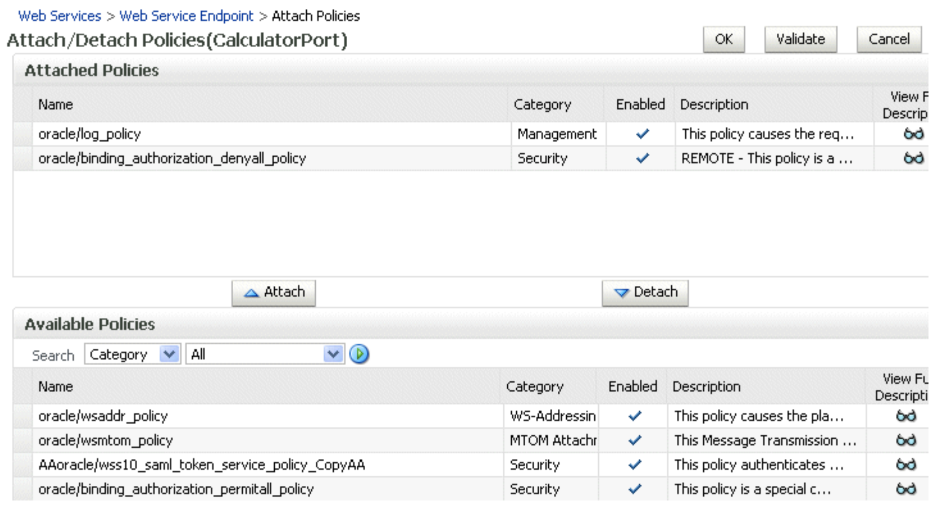
Follow this procedure to attach a policy to a single Web service. See "Attaching a Policy to Multiple Subjects (Bulk Attachment)" to attach a policy to multiple Web services at the same time.

To attach a policy to a Web service:

1. Navigate to the home page for the Web service, as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service ports if they are not already displayed.
3. Click the name of a port to navigate to the Web Service Endpoints page for a particular Web service.
4. Click the **Policies** tab.

   A list of the policies that are already attached to the port is displayed. For example, consider the policies shown in Figure 8–1.
5. Click **Attach/Detach**.
6. Select a policy from the Available Policies list, and click **Attach**. SeeFigure 8–2.

Figure 8–2 Attaching Policies to a Web Service


Web Services > Web Service Endpoint > Attach Policies

Attach/Detach Policies(CalculatorPort)

Attached Policies

Name	Category	Enabled	Description	View F Descrip
oracle/log_policy	Management	✓	This policy causes the req...	👓
oracle/binding_authorization_denyall_policy	Security	✓	REMOTE - This policy is a ...	👓

Available Policies

Search Category All

Name	Category	Enabled	Description	View Fu Descripti
oracle/wsaddr_policy	WS-Addressin	✓	This policy causes the pla...	👓
oracle/wsmtom_policy	MTOM Attachr	✓	This Message Transmission ...	👓
AAoracle/wss10_saml_token_service_policy_CopyAA	Security	✓	This policy authenticates ...	👓
oracle/binding_authorization_permitall_policy	Security	✓	This policy is a special c...	👓

7. Continue selecting and attaching policies. When you are finished, click **Validate** to verify that the combination of policies selected are valid.
8. Click **OK**.
9. The Web Service Port page now displays the attached policy on the **Policies** tab.
10. Restart the Web service application.

Attaching a Policy to a Web Service Using WLST

Use the following procedure to attach (or detach) a single policy, or multiple policies, to a single Web service port using WLST.

1. View the list of policies currently attached to the port as described in "Using WLST" in "Viewing the Policies That are Attached to a Web Service" on page 8-1.
2. View the list of available policies as described in "Displaying a List of the Available Policies Using WLST" on page 7-2.
3. To attach policies, do one of the following:
   - Use the `attachWebServicePolicy` command to attach a single policy to a Web service port. Specify the policy to be attached using the `policyURI` argument. If you specify a policy that is already attached or exists, then this command enables the policy if it is disabled.

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

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

     ```
     wls:/wls_domain/serverConfig> attachWebServicePolicy("/wls_
     domain/AdminServer/jaxwsejb30ws",
     "jaxwsejb","web","WsdlConcreteService","WsdlConcretePort",
     "oracle/wss_username_token_service_policy")
     ```

   - Use the `attachWebServicePolicies` command to attach multiple policies to a Web service port. Specify the policies to be attached using the policyURIs argument. If any of the policies that you specify in this command are already attached, then this command enables the policies that are already attached (if they are disabled), and attaches the others.

```
attachWebServicePolicies(application, moduleOrCompName, moduleType,
 serviceName, subjectName, policyURIs, [subjectType=None]
```

For example, to attach the policies `oracle/wss_username_token_service_policy` and `oracle/wsrm10_policy`to the `WsdlConcretePort` of the `WsdlConcreteService`, use the following command:

```
wls:/wls_domain/serverConfig> attachWebServicePolicies("/wls_
domain/AdminServer/jaxwsejb30ws",
"jaxwsejb","web","WsdlConcreteService","WsdlConcretePort",
["oracle/wss_username_token_service_policy","oracle/wsrm10_policy"])

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

Note: The policyURIs are validated through the Oracle WSM Policy Manager APIs if the wsm-pm application is installed on WebLogic Server and is available. If the policy validation fails, a message is displayed and the command is not executed.

If the wsm-pm application is not installed or is not available, these commands are not executed.

For additional information about validating policies, see "Validating Policy Subjects" on page 8-8.

---

4. To detach policies, do one of the following:

   - Use the `detachWebServicePolicy` command to detach a single policy from a Web service port. Specify the policy to be detached using the `policyURI` argument.

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

     For example, to detach the policy `oracle/wss_username_token_service_policy` from the `WsdlConcretePort` of the `WsdlConcreteService`, use the following command:

     ```
     wls:/wls_domain/serverConfig> detachWebServicePolicy("/wls_
     domain/AdminServer/jaxwsejb30ws",
     "jaxwsejb","web","WsdlConcreteService","WsdlConcretePort",
     "oracle/wss_username_token_service_policy")
     ```

   - Use the `detachWebServicePolicies` command to detach multiple policies from a Web service port. Specify the policies to be detached using the `policyURIs` argument.

     ```
     detachWebServicePolicies(application, moduleOrCompName, moduleType,
     serviceName, subjectName, policyURIs, [subjectType=None]
     ```

     For example, to detach the policies `oracle/wss_username_token_service_policy` and `oracle/wsrm10_policy`to the `WsdlConcretePort` of the `WsdlConcreteService`, use the following command:

```
wls:/wls_domain/serverConfig> detachWebServicePolicies("/wls_
domain/AdminServer/jaxwsejb30ws",
"jaxwsejb","web","WsdlConcreteService","WsdlConcretePort",
["oracle/wss_username_token_service_policy","oracle/wsrm10_policy"])

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

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

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

Attaching a Policy to Multiple Subjects (Bulk Attachment)

From the Application pages, you can attach one or more policies to one or more Web services.

Note: The bulk attachment mechanism does not perform validation on the policies that you attach.

The bulk attachment mechanism does not prevent you from creating an unsupported configuration such as having multiple authentication policies, or from attaching the same policy multiple times, and so forth.

Policy attachment is not synchronized automatically for ADF and WebCenter Web services in a cluster. When using ADF and WebCenter Web services in a cluster, you must attach and/or detach policies to each instance of the cluster. This issue does not apply to SOA composite applications.

To attach a policy to multiple Web services within an application:

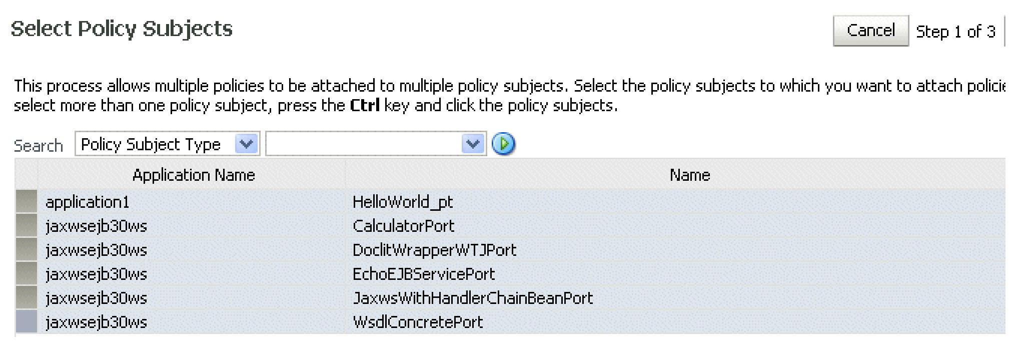
1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to attach the policy.
2. Select the domain, and then the instance of the server in which you want to attach the policy. The server can be an administration server or a managed server.
3. Using Fusion Middleware Control, click **WebLogic Server** and then **Web Services**.
4. From the Web Services Summary page, click **Attach Policies**.
5. From the Select Policy Subjects page, select one or more applications to which to attach a policy, as shown in Figure 8–3.

   Use the **Search** control to search for a particular policy subject type, a particular application name, or the type of Web service to which you want to attach a policy. Valid policy subject types include: Web Service Endpoint, Web Service Client, Web Service Connection, SOA Component, SOA Service, SOA Reference, or Asynchronous Callback Client. For more information about asynchronous callback clients, see "Developing Asynchronous Web Services" in *Concepts Guide to Oracle Infrastructure Web Services*.

   For example, if you choose to search for a policy subject type of Web Service Client, only available Web service clients, if any, are displayed.

   To select more than one application, press the Ctrl key and click the applications.

Figure 8–3 Select Subjects Page



Select Policy Subjects

Cancel | Step 1 of 3

This process allows multiple policies to be attached to multiple policy subjects. Select the policy subjects to which you want to attach polici
select more than one policy subject, press the **Ctrl** key and click the policy subjects.

Search: Policy Subject Type

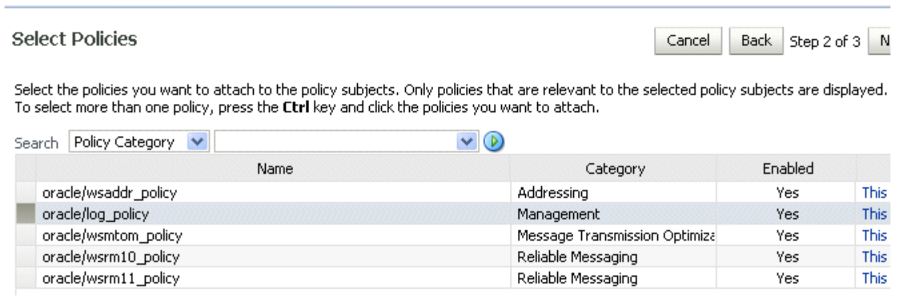
Application Name	Name
application1	HelloWorld_pt
jaxwsejb30ws	CalculatorPort
jaxwsejb30ws	DoclitWrapperWTJPort
jaxwsejb30ws	EchoEJBServicePort
jaxwsejb30ws	JaxwsWithHandlerChainBeanPort
jaxwsejb30ws	WsdlConcretePort

6. Click **Next.**

7. From the Select Policies page, select one or more policies that you want to attach to the selected applications, as shown in Figure 8–4. The Select Policies page shows only those policies that you can apply to all of the subjects selected in the previous step.

   To select more than one policy, press the Ctrl key and click the policies you want to attach.

Figure 8–4 Select Policies Page



Select Policies

Cancel | Back | Step 2 of 3 | N

Select the policies you want to attach to the policy subjects. Only policies that are relevant to the selected policy subjects are displayed.
To select more than one policy, press the **Ctrl** key and click the policies you want to attach.

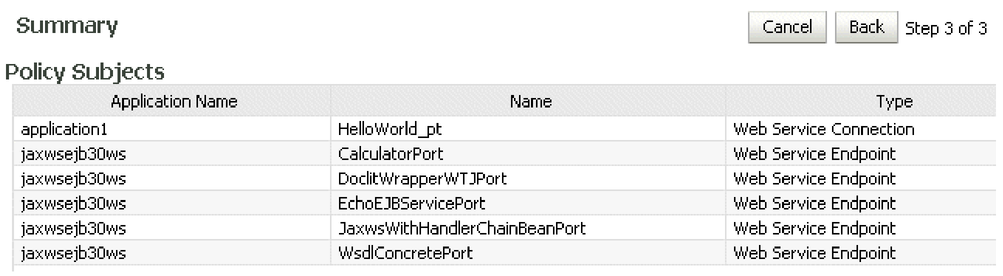
Search: Policy Category

Name	Category	Enabled	
oracle/wsaddr_policy	Addressing	Yes	This
oracle/log_policy	Management	Yes	This
oracle/wsmtom_policy	Message Transmission Optimiza	Yes	This
oracle/wsrm10_policy	Reliable Messaging	Yes	This
oracle/wsrm11_policy	Reliable Messaging	Yes	This

8. Click **Next.**

   The Summary page displays the applications you selected and the policies that will be attached to those applications, as shown in Figure 8–5.

Figure 8–5 Attachment Summary Page



Summary

Cancel | Back | Step 3 of 3

Policy Subjects

Application Name	Name	Type
application1	HelloWorld_pt	Web Service Connection
jaxwsejb30ws	CalculatorPort	Web Service Endpoint
jaxwsejb30ws	DoclitWrapperWTJPort	Web Service Endpoint
jaxwsejb30ws	EchoEJBServicePort	Web Service Endpoint
jaxwsejb30ws	JaxwsWithHandlerChainBeanPort	Web Service Endpoint
jaxwsejb30ws	WsdlConcretePort	Web Service Endpoint

9. Click **Back** to make any changes, or click **Attach** to complete the bulk attachment.

10. Restart the application that uses the Web services.

Validating Policy Subjects

The type and number of assertions within a policy may be valid and, therefore, a policy may be internally consistent and valid. However, when more than one policy is attached to a policy subject, the combination of policies must also be valid. Specifically, the following must be true:

- 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 Management 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 message protection can be attached to a subject.
- Only one security policy with subtype authorization can be attached to a subject.

> **Note:** There may be either one or two security policies attached to a policy subject. A security policy can contain an assertion that belongs to the authentication or message protection 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 an authentication policy and an authorization policy are both attached to a policy subject, the authentication policy must precede the authorization policy.
- 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.

You cannot use policy subject validation to check the validity of multiple policy subjects when you use the bulk attachment feature. After you attach the policies to your subjects with this feature, you must validate each subject individually.

> **Note:** The policy subject validation does not validate the XML schema of the policy. Therefore, if you manually edit the policy file, you must use another tool to check that the XML is valid.

To check for policy subject validation:

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

   The Application Deployment home page is displayed.

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

   This takes you to the Web Services summary page for your application.

3. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service ports if they are not already displayed.
4. Click the name of the port to navigate to the Web Service Endpoints page.
5. Click the **Policies** tab.
6. Click **Attach/Detach.**

7. Click **Validate.**

   If there is a validation error, a dialog box appears describing the error. Fix the error and do a policy subject validation again.

Attaching Policies to Web Service Clients

This section describes how to attach a policy to a Web service client, including SOA reference, ADF Data Control (DC), and asynchronous Web service Callback clients.

When using WLST to attach policies to a Web service client, the steps that you follow are also the same for all Web service client types. The argument settings specify the type of client to which you are attaching the policy.

Using Fusion Middleware Control

In Fusion Middleware Control, the steps you follow to attach a policy to a Web service client are the same for all Web service client types. However, how you navigate to the Web service client varies based on the application type, as described in the following sections.

Attaching Policies to SOA References

The following procedures describe how to attach policies to SOA references. For more information about developing SOA references, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

To attach policies to a SOA reference:

1. View the SOA reference, as described in "Viewing SOA References" on page 6-7.
2. Click the **Policies** tab.
3. Click **Attach/Detach**.
4. From the **Available Policies** portion of the page, select one or more policies that you want to attach. Click **Validate** to validate the policy, or Check Services Compatibility to make sure that the client policies are compatible with the service policies.
5. Click **Attach** when you are sure that you want to attach the policy or policies.
6. Click **OK**.

Attaching Policies to ADF DC Web Service Clients

The following procedure describes how to attach policies to an ADF DC Web service client. For more information about developing ADF DC Web service clients, see "Using Oracle ADF Model in a Fusion Web Application" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

To attach policies to an ADF DC Web service client:

1. Use Fusion Middleware Control to expand Application Deployments.
2. Click on the target application.
3. From the Application Deployments menu, select ADF.
4. Click on the Administration tab.
5. In the **Configure Properties** section of the page, click **ADF Connections**.

6. Select a row in the **Web Service Connections** list and click **Configure Web Service**.
7. From the Web Service Connection popup, select a Web service client to configure.
8. Click the **Policies** tab.
9. Click **Attach/Detach**.
10. From the **Available Policies** portion of the page, select one or more policies that you want to attach. Click **Validate** to validate the policy, or Check Services Compatibility to make sure that the client policies are compatible with the service policies.
11. Click **Attach** when you are sure that you want to attach the policy or policies.
12. Click **OK**.

Attaching Policies to Asynchronous Web Service Callback Clients

The following procedure describes how to attach policies to an asynchronous Web service Callback client. For more information about developing asynchronous Web services and callback clients, see "Developing Asynchronous Web Services" in *Concepts Guide to Oracle Infrastructure Web Services*.

To attach policies to an asynchronous Callback client:

1. Navigate to the endpoint for the asynchronous Web service, as described in "Viewing the Details for a Web Service Port" on page 6-5.
2. Click **Callback Client** in the upper right portion of the endpoint page.
3. Click the **Policy** tab.
4. Click **Attach/Detach**.
5. From the **Available Policies** portion of the page, select one or more policies that you want to attach. Click **Validate** to validate the policy, or Check Services Compatibility to make sure that the client policies are compatible with the service policies.
6. Click **Attach** when you are sure that you want to attach the policy or policies.
7. Click **OK**.

Using WLST

The following procedure describes how to attach policies to SOA references, ADF Web service DC and WebCenter clients, and asynchronous Web service callback clients. The steps that you follow are the same for each type of client. However, the argument settings will vary depending on the type of client to which you are attaching or detaching policies.

1. View the Web service clients as described in Using WLST in "Viewing Web Service Clients" on page 6-7.
2. Use the `listWebServiceClientPorts` command to display the port name and endpoint URL for a Web service client.

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

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

   ```
   wls:/wls-domain/serverConfig> listWebServiceClientPorts('/base_
   ```

```
domain/AdminServer/application1#V2.0',
'test1','wsconn','client')

HelloWorld_pt
```

3. View the list of available policies as described in "Displaying a List of the Available Policies Using WLST" on page 7-2.

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

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

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

   - Use the `attachWebServiceClientPolicy` command to attach a single policy to a Web service client port.

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

     Set the arguments as follows:

     – For a SOA reference, specify the name of the SOA composite using the `moduleOrCompName` argument, specify `soa` for the `moduleType` argument, and the name of the SOA reference using the `serviceRefName` argument.

     – For an ADF DC or WebCenter client, specify the name of the client application using the `application` argument, specify `wsconn` for the `moduleType` argument, and the service reference name using the `serviceRefName` argument.

     – For an asynchronous Web service callback client, specify `web` for the `moduleType` argument. Specify the name of the client application or SOA composite using the `application` and `moduleOrCompName` arguments, respectively.

     – For all client types, specify the name of the port using the `portInfoName` argument.

     – Specify the policy to be attached using the `policyURI` argument. If you specify a policy that is already attached or exists, then this command enables the policy if it is disabled.

     For example, to attach the client policy `oracle/wss_username_token_client_policy` to the `HelloWorld_pt` of the `client` service, use the following command:

     ```
     wls:/wls_domain/serverConfig> attachWebServiceClientPolicy("/wls_
     domain/AdminServer/application1#2.0",
     "test1","wsconn","client","HelloWorld_pt","oracle/wss_username_token_
     client_policy")
     ```

   - Use the `attachWebServiceClientPolicies` command to attach multiple policies to a Web service client port. Set the arguments as described for attaching a single client policy above, however you specify multiple policies to be attached using the `policyURIs` argument. If any of the policies that you specify in this command are already attached, then this command enables the policies that are already attached (if they are disabled), and attaches the others.

     ```
     attachWebServiceClientPolicies(application, moduleOrCompName,
     ```

```
moduleType, serviceRefName, portInfoName, policyURIs, [subjectType=None]
```

For example, to attach the policies `oracle/wss_username_token_client_policy` and `oracle/wsrm10_policy` to the `HelloWorld_pt` of the `client` service, use the following command:

```
wls:/wls_domain/serverConfig> attachWebServiceClientPolicies("/wls_
domain/AdminServer/application1#2.0",
"test1","wsconn","client","HelloWorld_pt",
["oracle/wss_username_token_client_policy","oracle/wsrm10_policy"])

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

Note: The policyURIs are validated through the Oracle WSM Policy Manager APIs if the wsm-pm application is installed on WebLogic Server and is available. If the policy validation fails, a message is displayed and the command is not executed.

If the wsm-pm application is not installed or is not available, these commands are not executed.

For additional information about validating policies, see "Validating Policy Subjects" on page 8-8.

---

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

   - Use the `detachWebServiceClientPolicy` command to detach a single policy from a Web service client port.

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

     Set the arguments as described in step 4 above.

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

     ```
     wls:/wls_domain/serverConfig> detachWebServiceClientPolicy("/wls_
     domain/AdminServer/application1#2.0",
     "test1","wsconn","client","HelloWorld_pt","oracle/wss_username_token_
     client_policy")
     ```

   - Use the `detachWebServiceClientPolicies` command to detach multiple policies from a Web service client port. Set the arguments as described for detaching a single client policy above, however you specify multiple policies to be detached using the `policyURIs` argument.

     ```
     detachWebServiceClientPolicies(application, moduleOrCompName,
     moduleType, serviceRefName, portInfoName, policyURIs, [subjectType=None]]
     ```

     For example, to detach the policies `oracle/wss_username_token_client_policy` and `oracle/wsrm10_policy` from the `HelloWorld_pt` of the `client` service, use the following command:

     ```
     wls:/wls_domain/serverConfig> detachWebServiceClientPolicies("/wls_
     domain/AdminServer/application1#2.0",
     "test1","wsconn","client","HelloWorld_pt",
     ["oracle/wss_username_token_client_policy","oracle/wsrm10_policy"])
     ```

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

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

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

Attaching Client Policies Permitting Overrides

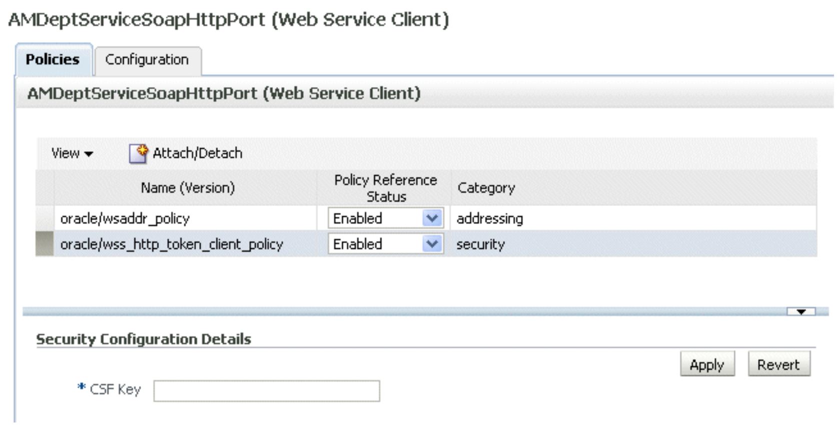
The policy configuration override feature allows you to specify certain Web service client configuration information on a per-client basis, in addition to or in lieu of setting it globally for any attachment of the policy. This targeting of configuration information limits the number of distinct policies you need to maintain.

You can define a single policy, and specify a default value for a configuration value. Rather than creating multiple policies with slightly varied configurations, you could use the same generic policy and override specific values to meet your requirements.

For example, the *oracle/wss_http_token_client_policy* policy is one example of a policy that includes the *csf-key* property, which has a default value of *basic.credentials*. The value signifies a key that maps to a username/password.   It might happen that you will always use the same key value any time you attach this policy to any number of Web service clients. In this case, you can specify the key value on the *oracle/wss_http_token_client_policy* policy **Configurations** page and have it apply to every instance.

However, you also have the option to override this key value on a per-client basis. After you attach a client policy that includes a property you can override, you can then supply a value in the **Security Configuration Setting** section of the **Policies** page, as shown in Figure 8–6.

Figure 8–6 Overriding a Configuration Property


AMDeptServiceSoapHttpPort (Web Service Client)

AMDeptServiceSoapHttpPort (Web Service Client)

Name (Version)	Policy Reference Status	Category
oracle/wsaddr_policy	Enabled	addressing
oracle/wss_http_token_client_policy	Enabled	security

Security Configuration Details

* CSF Key

You can override only the following properties in Web service client policies:

- *user.roles.include* (Optional, does not have to be set.)
- *csf-key* (Must be set on policy **Configuration** page or overridden.)
- *saml.issuer.name* (Optional, does not have to be set.)
- *saml.assertion.filename* (Optional, does not have to be set.)
- *service.principal.name* (Must be set on policy **Configuration** page or overridden.)

- *keystore.recipient.alias* (Can be set on policy **Configuration** page or overridden. Superseded by the "Using Service Identity Certification Extension" on page 9-10 feature. If the client overrides the *keystore.recipient.alias* property, the override is always used, and not the certificate published in the WSDL.)
- *keystore.sig.csf.key* (Optional, does not have to be set.)
- *keystore.enc.csf.key* (Optional, does not have to be set.)

Note: The keystore.enc.csf.key property puts the client's certificate in the replyTo header.

For WSS11 policies, keystore.enc.csf.key is used for asynchronous clients only.

For WSS10 policies, keystore.enc.csf.key is used for both asynchronous and synchronous clients.

Clearing a Configuration Property

If you need to clear an overridden configuration property, set it to an empty string.

Before you clear it, remember that other policies could be using the same property. The properties are client-specific and there could be multiple policies that are attached to the same client that use the same property.

Attaching Web Service Policies Permitting Overrides

You can specify a value for server-side configuration properties in a predefined or custom Web service policy, and then either use that value each time you attach the policy to a Web service or override it on a per-attachment basis.

For example, you might specify an IP address as a configuration property, and then validate the IP address from your Web service.

The scope for the server-side configuration property value is limited to the specific policy. That is, you could have two policies with the same server-side configuration property name, say *P1*, attached to the same Web service endpoint, and the two *P1* properties can have different values.

Server side properties that you can override are of two types: included in the predefined policies, and user defined.

- The server-side configuration properties included with the predefined policies allow you to override certain domain-wide configuration settings from a policy, such as the CSF key used for storing the signature-key password.
- For a user-defined property, you can add a property that has meaning in your environment. You can add a user-defined server-side property to the predefined policies, or to a custom policy.

Configuring Server-Side Override Properties for Message Protection Policies

This release of Oracle WSM includes the server-side override properties shown in Table 8–1 for message protection policies.

If you set (or then override) these properties, the new values are used in the attached Web service instead of the keystore passwords you configure as part of setting up the keystore for message protection, as described in "Setting up the Keystore for Message Protection" on page 9-7.

If you do not set these properties and leave the default blank values, the values you configure as part of setting up the keystore for message protection are used instead, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Table 8–1 Server-Side Configuration Properties for Message Protection Policies

Property Name	Default Value	Description
keystore.sig.csf.key	Blank	The alias and password used for storing the signature key password in the keystore. This property allows you to specify the signature key on a per-attachment level instead of at the domain level. (Applicable only to WSS10 message protection policies)
keystore.enc.csf.key	Blank	The alias and password used for storing the decryption key password in the keystore. This property allows you to specify the decryption key on a per-attachment level instead of at the domain level. (Applicable to WSS10 and WSS11 message protection synchronous policies)

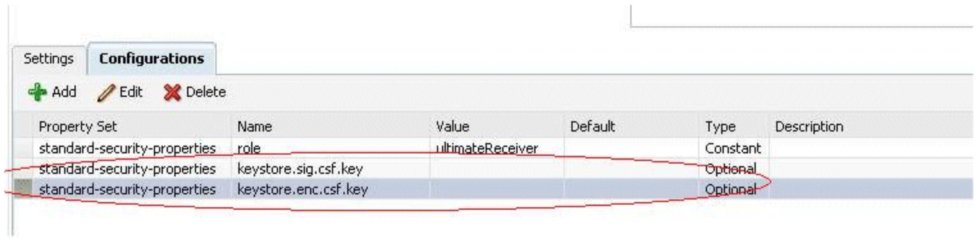
Setting Default Values for the Configuration Properties

By default, the *keystore.sig.csf.key* and *keystore.enc.csf.key* properties have a blank value. You can choose to set a value such that any Web service that attaches the policy can use these values, or override the values when you attach the policy.

To set a value of a configuration property for a policy:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select the message protection policy from the Policies table and click **Edit**.
3. On the Edit Policy page, click the **Configurations** tab.
4. Select the configuration property and click **Edit** to make the change to the *keystore.sig.csf.key* and *keystore.enc.csf.key* properties based on the keys in your keystore. See Figure 8–7.

Figure 8–7 Server-Side Configuration Properties



Property Set	Name	Value	Default	Type	Description
standard-security-properties	role	ultimateReceiver		Constant	
standard-security-properties	keystore.sig.csf.key			Optional	
standard-security-properties	keystore.enc.csf.key			Optional	

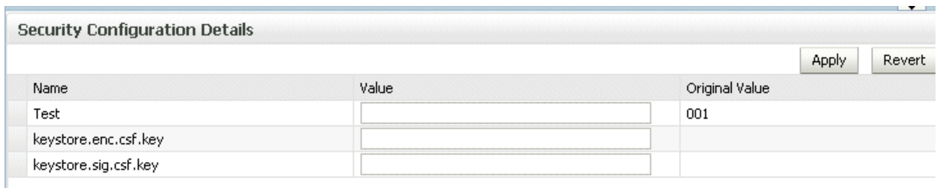
5. Validate your changes.
6. Click **Save**.

Overriding the Configuration Properties When Attaching a Policy

After you attach a message protection policy that includes a server-side configuration property, you can then override the existing value. In Fusion Middleware Control, you do this using the **Security Configuration Setting** section of the **Policies** page. In WLST, you do so using the `setWebServicePolicyOverride` command as described in "Overriding Configuration Properties When Attaching a Policy Using WLST" on page 8-20.

To do this using Fusion Middleware Control, select the attached policy with the server-side configuration property. The **Security Configuration Details** portion of the page appears, as shown in Figure 8–8.

Figure 8–8 Overriding a Server-Side Configuration Property



Name	Value	Original Value
Test		001
keystore.enc.csf.key		
keystore.sig.csf.key		

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* policy and that it is still blank. If Web service A attaches the *oracle/wss10_message_protection_service_policy* 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* policy attached to Web service A.

For all other policies, *keystore.sig.csf.key* has the value you configure as part of setting up the keystore for message protection are used instead, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Configuring Server-Side Override Properties for Authorization Policies

This release of Oracle WSM includes the server-side override properties shown in Table 8–2 for the *oracle/binding_permission_authorization_policy* policy. You can use these properties to set a different action and resource.

If you set (or then override) these properties, the new values are used in the attached Web service instead of the action and resource you configure as described in "How Authorization Permissions Are Determined" on page 10-45.

Table 8–2 Server-Side Configuration Property for Authorization Policies

Property Name	Default Value	Description
action	*	Specify the operations for which this policy should be enforced.
resource	*	Specify the Web service name (Namespace of Web service plus ServiceName)

Setting Default Values for the Configuration Properties

By default, the *action* and *resource* properties have a value of *.   You can choose to set a different value such that any Web service that attaches the policy can use that value, or override the value when you attach the policy.

To set a value for the configuration property:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select the *oracle/binding_permission_authorization_policy* policy from the Policies table and click **Edit**.
3. On the Edit Policy page, click the **Configurations** tab.
4. Select the configuration property and click **Edit** to make the change to the *action* or *resource* property based on your environment.
5. Validate your changes.
6. Click **Save**.

Overriding the Configuration Properties When Attaching a Policy

After you attach the *oracle/binding_permission_authorization_policy* policy, you can then override the existing value in the **Security Configuration Setting** section of the **Policies** page using Fusion Middleware Control. In WLST, you do so using the `setWebServicePolicyOverride` command as described in "Overriding Configuration Properties When Attaching a Policy Using WLST" on page 8-20.

To do this using Fusion Middleware Control, select the attached *oracle/binding_permission_authorization_policy* policy. The **Security Configuration Details** portion of the page appears.

Configuring User-Defined Client- or Server-Side Override Properties

You can use the Add New Configure Property feature to add one or more configuration properties that have meaning in your environment. Specifically, you can add one or more user-defined server- or client-side properties to the predefined policies, or to a custom policy. Then, you can either use the user-defined property as-is, or override it when you attach the policy.

In both cases, the property must already exist in the policy before you can override it when attaching the policy to a Web service or client. That is, you can override only those properties that are already present in the policy.

Therefore, you would typically add a user-supplied property with some default value to the predefined or custom policy, and then override it on a per-attachment basis.

You can add a user-defined property of type required, optional, or constant, but you cannot override a property of type constant.

Scope of User-Defined Configuration Properties

As with the predefined configuration properties, the scope for user-defined configuration properties in a policy differs for clients and Web services. Consider the following:

- The scope for a client-side configuration property value is the client. There could be multiple policies that are attached to the same client that use the same property.
- The scope for a server-side configuration property value is limited to the specific policy. That is, you could have two policies with the same server-side configuration property name, say *P1*, attached to the same Web service endpoint, and the two *P1* properties can have different values.

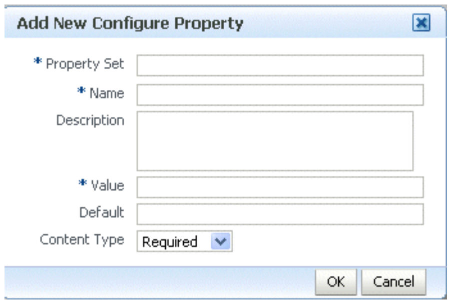
Adding a User-Defined Configuration Property

You edit the predefined or custom policy to add a user-defined configuration property.

To add a user-defined configuration property:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select the policy for which you want to add a property from the Policies table and click **Edit**.
3. On the Edit Policy page, click the **Configurations** tab.
4. Click Add. The Add New Configure Property dialog box shown in Figure 8–9 appears.

Figure 8–9 Adding a New Configuration Property


Add New Configure Property

* Property Set	
* Name	
Description	
* Value	
Default	
Content Type	Required

5. Enter the following information and click **OK**.
   - **Property Set** is your name for the group (set) to which you want this property to belong. This is a required field.
   - **Name** is your name for the property. The name must be unique for this policy. This is a required field.
   - **Description** is your description for the property.
   - **Value** is the current String value for the property. This is a required field.
   - **Default** is the default String value for the property if it is not otherwise set.
   - **Content Type** can be one of Constant, Optional, or Required. You can subsequently override only properties of type Optional and Required.
6. Validate the policy.
7. Click **Save**.

Editing a User-Defined Configuration Property

You can edit a user-defined configuration property if you need to change it.

To edit a user-defined configuration property:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select the policy for which you want to edit a property from the Policies table and click **Edit**.

3. On the Edit Policy page, click the **Configurations** tab.
4. Select the user-defined configuration property you want to edit and click **Edit**.
5. Make any needed changes.
6. Validate the policy.
7. Click **Save**.

Deleting a User-Defined Configuration Property

You can delete a user-defined configuration property if you no longer need it.

To delete a user-defined configuration property:

1. Navigate to the Web Services Policy page, as described in "Navigating to the Web Services Policies Page in Fusion Middleware Control" on page 7-2.
2. From the Web Services Policies page, select the policy for which you want to delete a property from the Policies table and click **Edit**.
3. On the Edit Policy page, click the **Configurations** tab.
4. Select the user-defined configuration property you want to delete and click **Delete**.
5. Validate the policy.
6. Click **Save**.

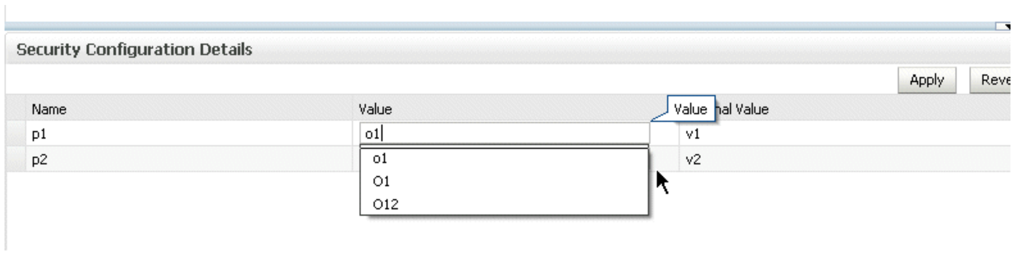
Overriding the Configuration Properties When Attaching a Policy

When you attach a policy that has a user-defined configuration property, you can override the existing value in the **Security Configuration Setting** section of the **Policies** page using Fusion Middleware Control. In WLST, you do so using the `setWebServicePolicyOverride` command as described in "Overriding Configuration Properties When Attaching a Policy Using WLST" on page 8-20.

To do this using Fusion Middleware Control, attach the policy as described in "Attaching a Policy to a Single Subject" on page 8-3, "Attaching a Policy to Multiple Subjects (Bulk Attachment)" on page 8-6, or "Attaching Policies to Web Service Clients" on page 8-9 as appropriate.

Then, select the attached policy with the user-defined property. The **Security Configuration Details** portion of the page appears, as shown in Figure 8–10.

Figure 8–10 Overriding a User-Defined Configuration Property



Name	Value	nal Value
p1	o1	v1
p2		v2

Overriding Configuration Properties When Attaching a Policy Using WLST

When you attach a policy that has an overridable property, you can override the existing value using the `setWebServicePolicyOverride` command. To do so, use the following procedure.

1. Attach the policy to the service as described in "Attaching a Policy to a Web Service Using WLST" on page 8-4.
2. Use the `setWebServicePolicyOverride` command to override policy properties.

   ```
   setWebServicePolicyOverride(application,moduleOrCompName,moduleType,
   serviceName,portName,policyURI,properties)
   ```

   You can override the properties listed in Table 8–1 and Table 8–2, and user-defined properties as described in "Configuring User-Defined Client- or Server-Side Override Properties" on page 8-17.

   For example, to override the ROLE property in the `oracle/wss_username_token_service_policy policy`, use the following command:

   ```
   wls:/wls-domain/serverConfig>setWebServicePolicyOverride
   ('/wls_domain/AdminServer/Jaxwsejb30ws','jaxwsejb',
   'web','WsdlConcreteService','WsdlConcretePort',
   "oracle/wss_username_token_service_policy",[("ROLE","ADMIN")])
   ```

   Notes: If the policy that you specify is not attached to the port, an error message is displayed and/or an exception is thrown.

   If you set the `properties` argument to `None`, then all policy overrides are removed.

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

For more information about this WLST command and its arguments, see "Web Services Custom WLST Commands" in *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

9

Setting Up Your Environment for Policies

This chapter describes how to set up your Fusion Middleware Control and WebLogic Server environments for security policies.

This chapter includes the following sections:

- "Setting up the Keystore for Message Protection" on page 9-7
- "Using Service Identity Certification Extension" on page 9-10
- "Configuring the Credential Store Provider" on page 9-12
- "Configuring an Authentication Provider in WebLogic Server" on page 9-14
- "Configuring the SAML and Kerberos Login Modules" on page 9-15
- "Configuring SAML" on page 9-16
- "Using Kerberos Tokens" on page 9-19
- "SAML Message Protection Use Case" on page 9-23

Configuring Keystores for SSL

If you want to use any of the policies listed in "Which Policies Require You to Configure SSL?" on page 9-2 or "Which Policies Require You to Configure Two-Way SSL?" on page 9-2, you must configure keystores for SSL.

SSL provides secure connections by allowing two applications connecting over a network to authenticate the other's identity and by encrypting the data exchanged between the applications.

Authentication allows a server, and optionally a client, to verify the identity of the application on the other end of a network connection. Encryption makes data transmitted over the network intelligible only to the intended recipient. A client certificate (two-way SSL) can be used to authenticate the user.

This section describes how to set up a Web service client and the WebLogic Server Web service container to send requests over SSL.

In order to use SSL in a Web service application, you need to:

- Configure the WebLogic Server keystore and SSL settings.
- Configure the Web service client keystore and SSL settings.

These steps are described in the sections that follow.

Which Policies Require You to Configure SSL?

The predefined policies that require you to configure SSL are as follows:

- oracle/wss_http_token_over_ssl_service_policy
- oracle/wss_http_token_over_ssl_client_policy
- oracle/wss_saml_token_bearer_over_ssl_server_policy
- oracle/wss_saml_token_bearer_over_ssl_client_policy
- oracle/wss_saml_token_over_ssl_service_policy
- oracle/wss_saml_token_over_ssl_client_policy
- oracle/wss_username_token_over_ssl_service_policy
- oracle/wss_username_token_over_ssl_client_policy

In addition, you can create a new policy that requires SSL by using the following templates:

- oracle/wss_http_token_over_ssl_service_template
- oracle/wss_http_token_over_ssl_client_template
- oracle/wss_saml_token_bearer_over_ssl_service_template
- oracle/wss_saml_token_bearer_over_ssl_client_template
- oracle/wss_saml_token_over_ssl_service_template
- oracle/wss_saml_token_over_ssl_client_template
- oracle/wss_username_token_over_ssl_service_template
- oracle/wss_username_token_over_ssl_client_template

See Appendix C, "Predefined Assertion Templates" and Appendix B, "Predefined Policies" for more information on these assertions and policies.

Which Policies Require You to Configure Two-Way SSL?

The predefined policies that require you to configure two-way SSL are as follows:

- oracle/wss_saml_token_over_ssl_client_policy
- oracle/wss_saml_token_over_ssl_service_policy
- oracle/wss_username_token_over_ssl_client_policy, when mutual authentication is selected.
- oracle/wss_username_token_over_ssl_service_policy, when mutual authentication is selected.
- oracle/wss_http_token_over_ssl_client_policy, when mutual authentication is selected.
- oracle/wss_http_token_over_ssl_service_policy, when mutual authentication is selected.

In addition, you can create a new policy that requires two-way SSL by using the following templates:

- oracle/wss_saml_token_over_ssl_client_template
- oracle/wss_saml_token_over_ssl_service_template

How to Configure a Keystore on WebLogic Server

Private keys, digital certificates, and trusted certificate authority certificates establish and verify identity and trust in the WebLogic Server environment.

This section briefly summarizes the steps that are required to configure the keystore in WebLogic Server. See the following two sources for complete information:

- *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help* for complete information, particularly the topic "Servers: Configuration: Keystores."
- *Oracle Fusion Middleware Securing Oracle WebLogic Server*, particularly *Configuring Identity and Trust*.

WebLogic Server is configured with a default identity keystore *DemoIdentity.jks* and a default trust keystore *DemoTrust.jks*. In addition, WebLogic Server trusts the certificate authorities in the cacerts file in the JDK. This default keystore configuration is appropriate for testing and development purposes. However, these keystores should not be used in a production environment.

To configure identity and trust for a server:

1. Obtain digital certificates, private keys, and trusted CA certificates from Sun Microsystem's 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 Sun's keytool utility, see the keytool-Key and Certificate Management Tool description at `http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/keytool.html`.

   You can also use the digital certificates, private keys, and trusted CA certificates provided by the WebLogic Server kit. The demonstration digital certificates, private keys, and trusted CA certificates should be used only in a development environment.

2. Create one keystore for identity and one for trust. The preferred keystore format is JKS (Java KeyStore).
3. Load the private keys and trusted CAs into the keystores.
4. In the left pane of the Console, expand Environment and select **Servers**.
5. Click the name of the server for which you want to configure the identity and trust keystores.
6. Select **Configuration**, and then **Keystores**.
7. In the Keystores field, select the method for storing and managing private keys/digital certificate pairs and trusted CA certificates. These options are available:

   - Custom Identity and Custom Trust: Identity and trust keystores you create.

- Demo Identity and Demo Trust: The demonstration identity and trust keystores, located in the ..*server**lib* directory and the JDK cacerts keystore, are configured by default. Use for development only.
- Custom Identity and Java Standard Trust: A keystore you create and the trusted CAs defined in the cacerts file in the *JAVA_HOME**jre**lib**security* directory.
- Custom Identity and Command Line Trust: An identity keystore you create and command-line arguments that specify the location of the trust keystore.

8. In the Identity section, define attributes for the identity keystore.
   - Custom Identity Keystore: The fully qualified path to the identity keystore.
   - Custom Identity Keystore Type: The type of the keystore. Generally, this attribute is Java KeyStore (JKS); if left blank, it defaults to JKS.
   - Custom Identity Keystore Passphrase: The password you will enter when reading or writing to the keystore. This attribute is optional or required depending on the type of keystore. All keystores require the passphrase in order 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 in order 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.

Configuring SSL on WebLogic Server (One-Way)

With one-way SSL, the server is required to present a certificate to the client but the client is not required to present a certificate to the server.

After you configure identity and trust keystores for a WebLogic Server instance as described in "Configuring Keystores for SSL" on page 9-1, 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 *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

To configure SSL:

1. In the left pane of the WebLogic Server Administration Console, expand Environment and select **Servers**.
2. Click the name of the server for which you want to configure SSL.
3. Select **Configuration**, and then the **SSL** page, and choose the location of identity (certificate and private key) and trust (trusted CAs) for WebLogic Server.
4. Set SSL attributes for the private key alias and password.
5. At the bottom of the page, click **Advanced**.
6. Set Hostname Verification to None.
7. Indicate the number of times WebLogic Server can use an exportable key between a domestic server and an exportable client before generating a new key. The more secure you want WebLogic Server to be, the fewer times the key should be used before generating a new key.
8. Set the Two Way Client Cert Behavior control to Client Certs Not Requested.
9. Specify the inbound and outbound SSL certificate validation methods. These options are available:
   - Builtin SSL Validation Only: Uses the built-in trusted CA-based validation. This is the default.
   - Built-in SSL Validation and Cert Path Validators: Uses the built-in trusted CA-based validation and uses configured CertPathValidator providers to perform extra validation.

Configuring SSL on WebLogic Server (Two-Way)

With two-way SSL, the server presents a certificate to the client and the client presents a certificate to the server. WebLogic Server can be configured to require clients to submit valid and trusted certificates before completing the SSL handshake.

After you configure identity and trust keystores for a WebLogic Server instance as described in "Configuring Keystores for SSL" on page 9-1, you can configure its two-way SSL attributes if the policy or template you are using requires it, as described in "Which Policies Require You to Configure Two-Way SSL?" on page 9-2.

This section summarizes the steps required to configure SSL on WebLogic Server. For complete information, see *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

To configure two-way SSL:

1. In the left pane of the WebLogic Server Administration Console, expand Environment and select **Servers**.
2. Click the name of the server for which you want to configure SSL.
3. Select **Configuration**, and then the **SSL** page, and choose the location of identity (certificate and private key) and trust (trusted CAs) for WebLogic Server.
4. Set SSL attributes for the private key alias and password.
5. At the bottom of the page, click **Advanced**.

6. Set Hostname Verification to None.
7. Indicate the number of times WebLogic Server can use an exportable key between a domestic server and an exportable client before generating a new key. The more secure you want WebLogic Server to be, the fewer times the key should be used before generating a new key.
8. Set the Use Server Certs control if needed. Setting this control determines whether a Web service client hosted on WebLogic Server should use the server certificates/key as the client identity when initiating a connection over HTTPS.
9. Set the **Two Way Client Cert Behavior** control to Client Certs Requested and Enforced.
10. Specify the inbound and outbound SSL certificate validation methods. These options are available:
    - Builtin SSL Validation Only: Uses the built-in trusted CA-based validation. This is the default.
    - Builtin SSL Validation and Cert Path Validators: Uses the built-in trusted CA-based validation and uses configured CertPathValidator providers to perform extra validation.

Configuring SSL for a Web Service Client

The core WebLogic Server security subsystem uses private key and X.509 certificate pairs, stored in the default keystores, for SSL.

You must ensure that the Web service client trusts the X.509 certificate that WebLogic Server uses to digitally sign the request. Do one of the following:

1. Ensure that WebLogic Server obtains a digital certificate that the client automatically trusts, because it has been issued by a trusted certificate authority.
2. Create a certificate registry that lists all the individual certificates trusted by WebLogic Server, and then ensure that the client trusts these registered certificates.

To configure SSL for a Web service client:

1. Create a keystore used by the client application. Oracle recommends that you create one client keystore per application user.

   You can use the keytool utility to perform this step. For development purposes, the keytool utility is the easiest way to get started.
2. Create a private key and digital certificate pair, and load it into the client keystore.

   Make sure that the certificate's key usage allows both encryption and digital signatures. Oracle requires a key length of 1024 bits or larger.
3. Make sure that the following properties are set in the client's JVM:
    - javax.net.ssl.trustStore -- The name of the file that contains the trust store.
    - javax.net.ssl.trustStoreType -- The type of KeyStore object that you want the default TrustManager to use.
    - javax.net.ssl.trustStorePassword -- The password for the KeyStore object that you want the default TrustManager to use.

Configuring 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. Do one of the following:

1. Ensure that the client application obtains a digital certificate that WebLogic Server automatically trusts, because it has been issued by a trusted certificate authority.
2. Create a certificate registry that lists all the individual certificates trusted by WebLogic Server, and then ensure that the client uses one of these registered certificates.

To configure SSL for a Web service client:

1. Create a keystore used by the client application. Oracle recommends that you create one client keystore per application user.

   You can use the keytool utility to perform this step. For development purposes, the keytool utility is the easiest way to get started.
2. Create a private key and digital certificate pair, and load it into the client keystore.

   Make sure that the certificate’s key usage allows both encryption and digital signatures. Oracle requires a key length of 1024 bits or larger.
3. Make sure that the following properties are set in the client's JVM:
   - javax.net.ssl.trustStore -- The name of the file that contains the trust store.
   - javax.net.ssl.trustStoreType -- The type of KeyStore object that you want the default TrustManager to use.
   - javax.net.ssl.trustStorePassword -- The password for the KeyStore object that you want the default TrustManager to use.
   - javax.net.ssl.keyStore -- The name of the file that contains the KeyStore object.
   - javax.net.ssl.keyStoreType -- The type of KeyStore object.
   - javax.net.ssl.keyStorePassword -- The password for the KeyStore.

Setting up the Keystore for Message Protection

In order to sign and encrypt SOAP messages you must first create and configure the Web Services Manager Keystore for a WebLogic domain. This keystore is used to store public and private keys for SOAP messages within the WebLogic Domain.

Note: The Web services manager runtime does **not** use the WebLogic Server keystore that is used for SSL as documented elsewhere in this chapter.

The signature and encryption keys are used to sign, verify, encrypt, and decrypt the SOAP messages.

The keystore configuration is domain wide: all Web services and Web service clients in the domain use this keystore.

To set up the keystore used by Web Services Manager follow these steps:

1. Use the keytool to create a Java keystore, as described in "How to Create and Use a Java Keystore" on page 9-9.

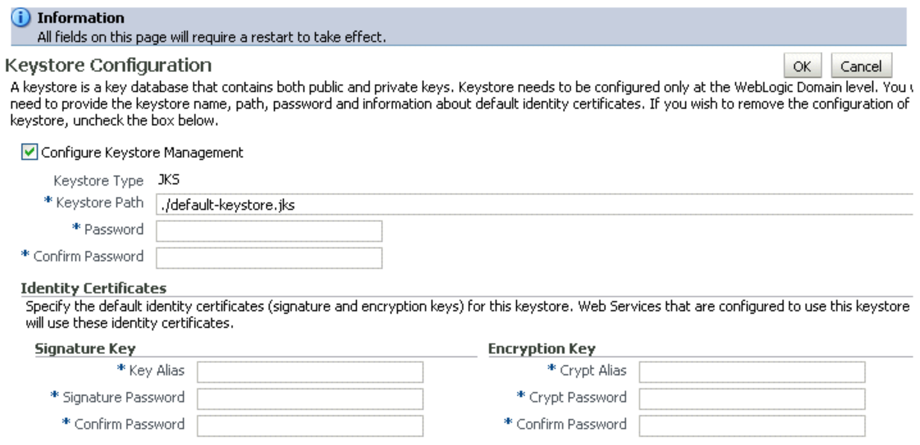
2. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure the keystore. Select the domain.

3. Using Fusion Middleware Control, click **Weblogic Domain**, then **Security**, and then **Security Provider Configuration**.

   Click the plus sign (+) to expand the **Keystore** control near the bottom of the page, then click **Configure**.

   The Web Services Manager Keystore Configuration page is displayed, as shown in Figure 9–1.

Figure 9–1 Web Services Manager Keystore Configuration



Information
All fields on this page will require a restart to take effect.

Keystore Configuration

OK Cancel

A keystore is a key database that contains both public and private keys. Keystore needs to be configured only at the WebLogic Domain level. You u
need to provide the keystore name, path, password and information about default identity certificates. If you wish to remove the configuration of
keystore, uncheck the box below.

[✓] Configure Keystore Management

Keystore Type	JKS
* Keystore Path	./default-keystore.jks
* Password	
* Confirm Password	

Identity Certificates
Specify the default identity certificates (signature and encryption keys) for this keystore. Web Services that are configured to use this keystore will use these identity certificates.

Signature Key		Encryption Key	
* Key Alias		* Crypt Alias	
* Signature Password		* Crypt Password	
* Confirm Password		* Confirm Password	

4. If it is not already enabled, click the Configure Keystore Management check box.

5. Enter the path and name for the keystore that you created. By default, the keystore name is *default-keystore.jks*, but you can change this. However, you cannot change the keystore type; it must be JKS.

6. Enter a password for the keystore and confirm it.

7. Enter an alias and password for the signature and encryption keys. Confirm the passwords.

   The alias and password for the signature and encryption keys define the string alias and password used to store and retrieve the keys.

8. Click OK to submit the changes.

   Note that all fields on this page require a restart of Oracle Enterprise Manager Fusion Middleware Control to take effect.

Note: The Oracle WSM agent caches the keystore name and object. If you make subsequent changes to the contents of the keystore or to its name, you must restart Oracle Enterprise Manager Fusion Middleware Control.

Setting Up the Web Service Client Keystore at Design Time

You need to create a Java Key Store (JKS) keystore to store the signature and encryption keys required by the X.509 token on the client. Keys are used for a variety of purposes, including authentication and data integrity. For example:

- To sign data, you must have the signer's private key.
- To verify a signature, you must have a trusted CA certificate and the public key that matches the private key.
- To encrypt data, you must have the recipient's public key. The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.
- To decrypt data, you must have the private key that corresponds to the public key.

These trusted certificates and public and private keys are stored in the keystore. The following sections describe where you can obtain trusted certificates and how to create and use these keystores.

- "How to Obtain a Trusted Certificate" on page 9-9
- "How to Create and Use a Java Keystore" on page 9-9
- "How to Create Private Keys and Load Trusted Certificates" on page 9-9

How to Obtain a Trusted Certificate

You can obtain a certificate from a Certificate Authority (CA), such as Verisign or Entrust, 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.

How to Create and Use a Java Keystore

The Java Keystore (JKS) is the proprietary keystore format defined by Sun Microsystems. To create and manage the keys and certificates in the JKS, use the keytool utility. You can use the keytool utility to perform the following tasks:

- Create public and private key pairs, designate public keys belonging to other parties as trusted, and manage your keystore.
- Issue certificate requests to the appropriate Certification Authority (CA), and import the certificates which they return.
- Administer your own public and private key pairs and associated certificates. This allows you to use your own keys and certificates to authenticate yourself to other users and services. This process is known as "self-authentication." You can also use your own keys and certificates for data integrity and authentication services, using digital signatures.
- Cache the public keys of your communicating peers. The keys are cached in the form of certificates.

How to Create Private Keys and Load Trusted Certificates

The following section provides an outline of how to create and manage the JKS with the keytool utility. It describes how to create a keystore and to load private keys and trusted CA certificates. You can find more detailed information on the commands and arguments for the keytool utility at this Web address.

http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html

1. Create a new private key and self-signed certificate.

Use the genKey command to create a private key. It will create a new private key if one does not exist. The following command generates an RSA key, with RSA-SHA1 as the signature algorithm, with the alias test in the test.jks keystore.

keytool -genkey -alias test -keyalg "RSA" -sigalg "SHA1withRSA" -dname "CN=test, C=US" -keystore test.jks

The keytool utility prompts for the needed key and keystore passwords. DSA key is not supported. Make sure you pass the parameter " -keyalg RSA " in the command.

2. Display the keystore.

   The following command displays the contents of the keystore. It will prompt you for the keystore password.

   keytool -list -v -keystore test.jks

3. Import a trusted CA certificate in the keystore.

   Use the -import command to import the certificate. The following command imports a trusted CA certificate into the test.jks keystore. It will create a new keystore if one does not exist. The keytool utility prompts for the needed password.

   keytool -import -alias aliasfortrustedcacert -trustcacerts -file trustedcafilename -keystore test.jks

4. Generate a certificate request.

   Use the -certreq command to generate the request. The following command generates a certificate request for the test alias. The CA will return a certificate or a certificate chain.

   keytool -certreq -alias test -sigalg "RSAwithSHA1" -file certreq_file -storetype jks -keystore test.jks

5. Replace the self-signed certificate with the trusted CA certificate.

   You must replace the existing self-signed certificate with the certificate from the CA. To do this, use the -import command. The following command replaces the trusted CA certificate in the test.jks keystore. The keytool utility prompts for the needed password.

   keytool -import -alias test -file trustedcafilename -keystore test.jks

Using Service Identity Certification Extension

In prior releases of Oracle WSM, for Web services that implemented a message-protection policy the Web service client needed to store the Web service's public certificate in its domain-level keystore. The client then used the *keystore.recipient.alias* property to identify the certificate in the keystore. To do this, you either identified the *keystore.recipient.alias* property on the Configurations page or overrode it on a per-client basis using the Security Configuration Details control when attaching the policy (or programmatically).

In this release of Oracle WSM, for Web services that implement a message-protection policy the Web service's base64-encoded public certificate is published in the WSDL. The certificate is included for message protection policies whether or not the policy encrypts or decrypts data.

The certificate in the WSDL is the service's public key by default, as determined by the Encryption Key you specified when "Setting up the Keystore for Message Protection" on page 9-7.

If this certificate is not found, 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 in order to be trusted.

Hostname Verification Included in WSDL

This release includes a hostname verification feature that ensures that a certificate retrieved from a WSDL was not the subject of a substitution attack or "man in the middle" attack and is indeed the expected certificate.

To to this, Oracle WSM validates that the common name (CN) or the subject Group Base Distinguished Name (DN) in the certificate matches the hostname of the service.

This feature depends upon the subject DN of the certificate.

By default, hostname verification is disabled.

Enabling or Disabling Service Identity Certificate Extension and Hostname Verification

You use Fusion Middleware Control to enable or disable service identity certificate extension and hostname verification.

Service identity certificate extension is enabled by default; hostname verification is disabled by default.

Note: Service identity certificate extension does not set the encryption key from which the public key is derived. You must first specify this key as described in "Setting up the Keystore for Message Protection" on page 9-7.

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 "Setting up the Keystore for Message Protection" on page 9-7.

   If you use a service side override to override the encryption key or keystore for a Web service, the certificate corresponding to the overridden key is used.

2. From the navigation pane, expand **WebLogic Domain**.
3. Select the domain in which you want to enable or disable service identity certificate extension and hostname verification.
4. Using Fusion Middleware Control, click **WebLogic Domain**.
5. Select **Web Services**, and then select **Platform Policy Configuration**.
6. Select the **Identity Extension** tab.
7. Two controls are available:
   - Enable Identity Verification sets the *wsm.ignore.identity.wsdl* property to turn identity verification on and off. The default is on (false).

- Enable Hostname Verification sets the *wsm.ignore.hostname.verification* property to turn hostname verification on and off. The default is off (true).

Ignoring the Service Identity Certificate Extension From the Client

---

Note: If the client overrides the *keystore.recipient.alias* property (*oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_RECIPIENT_KEY_ALIAS* in a client programmatic override), the override is always used, and not the certificate published in the WSDL.

For a JEE client, the value of the wsm.ignore.identity.wsdl property is read automatically and no additional configuration is required. Set this property in Fusion Middleware Control to turn identity verification on and off, as described in "Enabling or Disabling Service Identity Certificate Extension and Hostname Verification" on page 9-11.

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");
```

Ignoring Hostname Verification from the Client

For a JEE client, the value of the wsm.ignore.hostname.verification property is read automatically and no additional configuration is required. Set this property in Fusion Middleware Control to turn hostname verification on and off, as described in "Enabling or Disabling Service Identity Certificate Extension and Hostname Verification" on page 9-11.

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");
```

Configuring the Credential Store Provider

The credential store provider provides a way to store, retrieve, and delete credentials for a Web service and other applications.

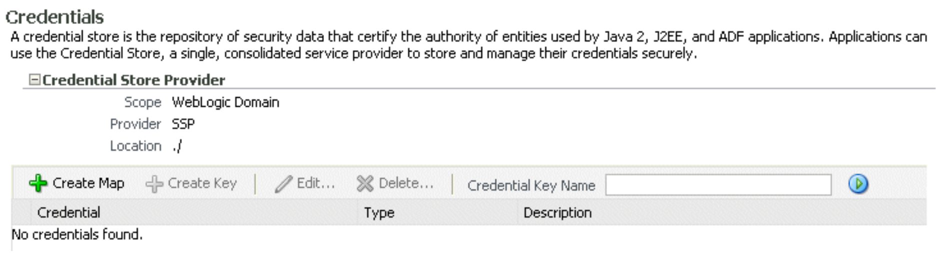
For example, the oracle/wss_http_token_client_policy policy includes the *csf-key* property, with a default value of *basic.credentials*. This credential is stored in the credential store provider.

Follow these steps to configure the credential store provider:

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure the keystore. Select the domain.
2. Using Fusion Middleware Control, click **Weblogic Domain**, then **Security**, and then **Credentials**.

The Credential Store Provider Configuration page is displayed, as shown in Figure 9–2. (In this figure, the **Credential Store Provider** control has already been expanded.)

Figure 9–2 Credential Store Provider Configuration Page


Credentials

A credential store is the repository of security data that certify the authority of entities used by Java 2, J2EE, and ADF applications. Applications can use the Credential Store, a single, consolidated service provider to store and manage their credentials securely.

Credential Store Provider

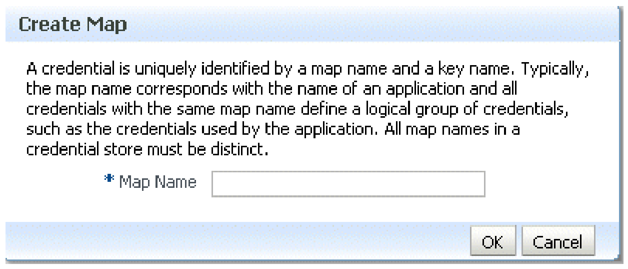
Scope	WebLogic Domain
Provider	SSP
Location	./

Credential	Type	Description

No credentials found.

3. Click **Create Map** and enter the map name *oracle.wsm.security*, as shown in Figure 9–3.

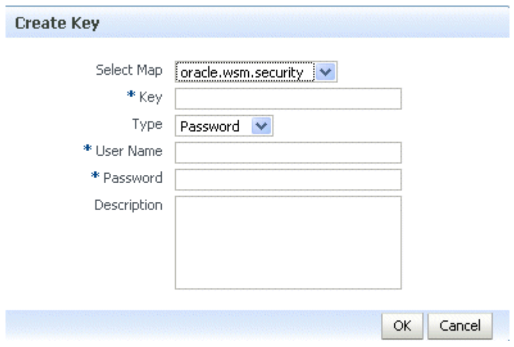
Figure 9–3 Set Security Provider Screen


Create Map

A credential is uniquely identified by a map name and a key name. Typically, the map name corresponds with the name of an application and all credentials with the same map name define a logical group of credentials, such as the credentials used by the application. All map names in a credential store must be distinct.

4. Click **Create Key**. The Create Key dialog box appears, as shown in Figure 9–4.

Figure 9–4 Create Key Dialog Box


Create Key

Select Map	oracle.wsm.security
* Key	
Type	Password
* User Name	
* Password	
Description	

5. Select the map name *oracle.wsm.security* if it is not already selected.
6. Enter the key name.
7. Select the key type, either *Password* or *Generic*. A password credential can store a username and password. A generic credential can store any credential object.
8. For a password credential, enter the username and password.
9. Click **OK**.

Configuring an Authentication Provider in WebLogic Server

This section introduces WebLogic Server security features that are described in detail in *Oracle Fusion Middleware Securing Oracle WebLogic Server* and in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*. This section provides only a brief introduction to the security features, and concentrates on how they relate to configuring policies.

The security policies that you use determine what types of security providers you must configure in WebLogic Server. You can categorize the policies based on their token type:

- Policies that use the username token require an authentication provider that can handle the *NameCallback* and *PasswordCallback*. The WebLogic Default Authentication provider is one such provider.

  The following policies fall into this category:

  - oracle/wss_http_token_service_policy
  - oracle/wss_username_token_service_policy
  - oracle/wss_username_token_over_ssl_service_policy
  - oracle/wss11_username_token_with_message_protection_service_policy
  - oracle/wss10_username_token_with_message_protection_service_policy
  - oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy
- Policies that use the X.509 and SAML tokens require an authentication provider (or Identity Assertion provider) that can handle perimeter authentication via the *NameCallback*. The Web service runtime process the tokens on your behalf to determine the username, and then invokes the Oracle Platform Security Service (OPSS) layer to complete the authentication. In this way, the security providers do not handle the X.509 or SAML tokens directly, and the WebLogic providers do not have to support these token types.

  The following policies fall into this category:

  - oracle/wss10_x509_token_with_message_protection_service_policy
  - oracle/wss10_saml_token_service_policy
  - oracle/wss10_saml_token_with_message_protection_service_policy
  - oracle/wss_saml_token_over_ssl
  - oracle/wss_saml_token_bearer_over_ssl_service_policy
  - oracle/wss10_saml_hok_token_with_message_protection_service_policy
  - oracle/wss11_saml_token_with_message_protection_service_policy
  - oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy
  - oracle/wss11_x509_token_with_message_protection_service_policy

What Type of WebLogic Security Authentication Providers Must You Create?

You can use any Weblogic Authentication provider that can validate the credentials in the *NameCallback* and *PasswordCallback* callbacks, or the *NameCallback* alone, as appropriate. This means that you can use the WebLogic Default Authentication provider and authenticate the user against the embedded LDAP datastore if you so choose, or the Default Identity Asserter, and so forth See "Configure Authentication and Identity Assertion Providers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help* for information on how to do this.

Configuring the SAML and Kerberos Login Modules

The SAML and Kerberos policies have associated login modules, as determined by the assertions that make up the policy. When you attach a SAML policy to a Web service, you must edit the login policy and make any needed changes. The Kerberos login module has settings that you can optionally configure.

(Login modules associated with other policy types do not have settings specific to the Web service policies.)

Table 9–1 lists the available login modules and which policies use them.

Table 9–1 SAML and Kerberos Login Modules and Related Policies

Login Module Service Name	Description	Settable Attributes and Values
saml.loginmodule	The SAML login module is a Java Authentication and Authorization Service (JAAS) login module that accepts SAML assertions to do a login. The SAML login module enables the Web services to be run using the login context of the principal created from the SAML assertion.	Issuers. Name of the issuer of the SAML token. www.oracle.com is the default.

Table 9–1 (Cont.) SAML and Kerberos Login Modules and Related Policies

Login Module Service Name	Description	Settable Attributes and Values
krb5.loginmodule	Kerberos login module	principal. The name of the principal that should be used. It could be simple username such as "testuser" or a service name such as "host/testhost.eng.sun.com" . You can use principal option to set the principal when there are credentials for multiple principals in the keyTab or when you want a specific ticket cache only.
		useKeyTab. True or false. Set this to true if you want the module to get the principal's key from the keytab (default value is False). If keytab is not set, then the module will locate the keytab from the Kerberos configuration file. If it is not specified in the Kerberos configuration file then it will look for the file {*user.home*}{*file.separator*}krb5.keytab.
		storeKey. Set this to True to if you want the principal's key to be stored in the Subject's private credentials.
		keyTab. Set this to the file name of the keytab to get principal's secret key.
		doNotPrompt. Set this to true if you do not want to be prompted for the password if credentials cannot be obtained from the cache or keytab (default is false). If set to true, authentication will fail if credentials cannot be obtained from the cache or keytab.

Do the following to configure a login module:

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you need to configure the keystore. Select the domain.
2. Using Fusion Middleware Control, click **Weblogic Domain**, then **Security**, and then **Security Provider Configuration**.
3. From the list of login modules, select a login module and click **Edit**.
4. Configure any specific attributes or custom properties for the login module.

Configuring SAML

The SAML standard defines a common XML framework for creating, requesting, and exchanging security assertions between software entities on the Web. The SAML Token profile is part of the core set of WS-Security standards, and specifies how SAML assertions can be used for Web services security. SAML also provides a standard way to represent a security token that can be passed across the multiple steps of a business process or transaction, from browser to portal to networks of web services.

If you use any of the following predefined policies, you must configure SAML:

- oracle/wss_saml_token_bearer_over_ssl_server_policy
- oracle/wss_saml_token_bearer_over_ssl_client_policy
- oracle/wss_saml_token_over_ssl_service_policy
- oracle/wss_saml_token_over_ssl_client_policy
- oracle/wss10_saml_token_service_policy
- oracle/wss10_saml_token_client_policy
- oracle/wss10_saml_token_with_message_protection_client_policy
- oracle/wss10_saml_token_with_message_protection_service_policy
- oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy
- oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy
- oracle/wss10_saml_hok_token_with_message_protection_service_policy
- oracle/wss10_saml_hok_token_with_message_protection_client_policy
- oracle/wss10_saml_token_with_message_integrity_service_policy
- oracle/wss10_saml_token_with_message_integrity_client_policy
- oracle/wss11_saml_token_with_message_protection_service_policy
- oracle/wss11_saml_token_with_message_protection_client_policy

How the SAML Token is Validated

The SAML login module verifies the SAML tokens on behalf of the Web service. The SAML login module then extracts the username from the verified token and (indirectly) passes it to Oracle Platform Security Services (OPSS) via the *NameCallback* to complete the perimeter authentication.

Which Authentication Provider is Used?

Any configured authentication provider (identity asserter) that handles the *NameCallback* can then be invoked.

The authentication provider then simply checks whether the user exists (identity assertion mode) and, if it does, the user is asserted and a subject is established.

How to Configure SAML Web Service Client at Design Time

Follow the steps described in this section to configure the SAML Web service client at design time. (If you attach the SAML policies to the Web service client at deploy time, you do not need to configure these properties and they are not exposed in Fusion Middleware Control.)

You can also include user roles in the assertion and change the SAML assertion issuer name, as described in subsequent sections.

Configure the Username for the SAML Assertion

For a JSE client application, configure the username as a BindingProvider property:

```
Map<String,Object>  reqContext = ((BindingProvider) proxy).getRequestContext()
   reqContext.put( BindingProvider.USERNAME_PROPERTY, "jdoe")
```

where *proxy* refers to the Web service proxy used for invoking the actual Web service.

For a JEE 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 JEE application and you are making a Web service call from that JEE 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.

Including User Roles in the Assertion

You can pass the user's role as an attribute statement in the SAML assertion. To do this at post-deploy time, configure the *user.role.include* property to "true." The default value in the policy is "false."

To configure the user's role at design time, set the *user.role.include* property to "true" in the BindingProvider.

How to Configure Oracle Platform Security Services (OPSS) for SAML Policies

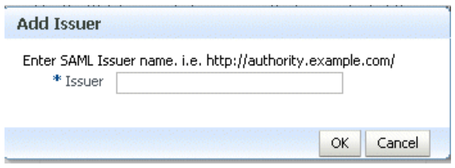
Follow these steps to configure OPSS for the predefined SAML policies:

1. Configure the SAML login module, as described in "Configuring the SAML and Kerberos Login Modules" on page 9-15.

   By default, the SAML assertion issuer name is *www.oracle.com*. The *saml.issuer.name* 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.

   To use a different issuer name, click **Add** to add an additional issuer name, as shown in Figure 9–5.

Figure 9–5 Adding a SAML Issuer to the Login Module



2. Configure the identity assertion provider in the WebLogic Server Administration Console.

3. If you will be using policies that involve signatures related to SAML assertions (for example, SAML holder-of-key policies) where a key referenced by the assertion is used to sign the message, or sender-vouches policies where the sender's key is used to sign the message, you need to configure keys and certificates for signing and verification, as described in "Setting up the Keystore for Message Protection" on page 9-7.

4. If you will be using policies that require SSL, you need to configure SSL, as described in "Configuring Keystores for SSL" on page 9-1.

Adding an Additional SAML Assertion Issuer Name

The *saml.issuer.name* property is generally *www.oracle.com* if you are using the predefined SAML policies (or assertions) on both the Web service client and Web service sides. Therefore, you can generally use the defaults and not configure any issuer.

If a different client, for instance .NET/WLS, is talking to a Web service protected by a predefined SAML policy, or you otherwise need to use a different SAML issuer, then you need to add an additional issuer name.

To add an additional SAML assertion issuer name:

1. Configure the SAML login module, as described in "Configuring the SAML and Kerberos Login Modules" on page 9-15. Click **Add** to add an additional issuer name, as shown in Figure 9–5.
2. At deploy time, specify a value for *saml.issuer.name* on the **Configurations** page for the SAML client policy, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The default value in the policy is *www.oracle.com*.

   To configure the issuer at design time, set the *saml.issuer.name* property in the BindingProvider.

Using Kerberos Tokens

Oracle Fusion Middleware 11*g* Release 1 (11.1.1) provides support for Kerberos tokens with the following predefined policies:

- oracle/wss11_kerberos_token_client_policy
- oracle/wss11_kerberos_token_service_policy
- oracle/wss11_kerberos_token_with_message_protection_client_policy
- oracle/wss11_kerberos_token_with_message_protection_service_policy

You may also create a policy using the following assertion templates:

- oracle/wss11_kerberos_token_client_template
- oracle/wss11_kerberos_token_service_template
- oracle/wss11_kerberos_token_with_message_protection_client_template
- oracle/wss11_kerberos_token_with_message_protection_service_template

See Appendix C, "Predefined Assertion Templates" and Appendix B, "Predefined Policies" for more information on these assertions and policies.

Configuring the KDC

Follow the steps described in this section to configure the KDC for use by the Web service client and Web service.

Initializing and Starting the KDC

Initialize KDC database. For example, on UNIX you might run the following command as root, where *oracle.com* is your default realm:

```
root# /usr/kerberos/sbin/kdb5_util -r oracle.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 &
```

Creating Principals

Create two accounts in the KDC user registry. The first account is for the end user; that is, the Web service client principal. The second account is for the Web service principal.

One way to create these accounts is with the kadmin.local tool, which is typically provided with MIT KDC distributions. For example:

```
>sudo su - # become root
>cd /usr/kerberos/sbin/kadmin.local
>kadmin.local>addprinc fmwadmin -pw welcome1
>kadmin.local> addprinc SOAP/myhost.oracle.com -randkey
>kadmin.local>listprincs # to see the added principals
```

The Web service principal name (SOAP/myhost.oracle.com) is shown in the example as being created with a random password. The Web service principals use keytables (a file that stores the service principal name and key) to log into Keberos System. Using a random password increases security.

Configuring the Web Service Client to Use the Correct KDC

The Web service client needs to be configured to authenticate against the right KDC.

The configuration for the KDC resides at */etc/krb5.conf* for UNIX hosts, and at *C:\windows\krb5.ini* for Windows hosts.

A sample krb5.conf is shown in Example 9–1. 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_enctypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc
  - default_tgs_enctypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc
  - permitted_enctypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc
- The order of cipher suites is significant. For Keberos message protection to work, the first in the list needs to "des3-cbc-sha1". This is because Oracle WSM supports the encryption algorithm TripleDES, but not plain DES.

Example 9–1 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 = oracle.com
dns_lookup_realm = false
dns_lookup_kdc = false
default_tkt_enctypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc
```

```
default_tgs_enctypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc
permitted_enctypes = des3-cbc-sha1 des-cbc-md5 des-cbc-crc

[realms]
oracle.com =
{kdc = someadminserver.com:88  admin_server = someadminserver.com:749


default_domain = us.oracle.com  }
[domain_realm]
us.oracle.com = oracle.com

[kdc]
profile = /var/kerberos/krb5kdc/kdc.conf

[appdefaults]
pam =
{   debug = false    ticket_lifetime = 36000   renew_lifetime = 36000


forwardable = true    krb4_convert = false  }
```

Setting the Service Principal Name In the Web Service Client

The Web service client that is enforcing Kerberos client side policies needs to know the service principal name of the service it is trying to access. You set the service principal name in "Creating Principals" on page 9-20.

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@oracle.com*.

Setting the Service Principal Name In the Web Service Client at Design Time

The Web service client that is enforcing Kerberos client side policies needs to know the service principal name of the service it is trying to access. You set the service principal name in "Creating Principals" on page 9-20.

Use a configuration override to specify the service principal name at design time, as follows:

```
JAX-WS Clients:
((BindingProvider)port).getRequestContext().put(SecurityConstants.ClientConstants.
WSSEC_KERBEROS_SERVICE_PRINCIPAL,
SOAP/myhost.oracle.com@oracle.com);
```

Configuring the Web Service to Use the Right KDC

Configure the Web service 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 KDC configuration for a Web service client is shown in Example 9–1. This example also applies to the Web service KDC configuration.

Using the Correct Keytab File in Enterprise Manager

To use the correct keytab file, you

- Extract and install the keytab File
- Modify the krb5 login module

These tasks are described in the sections that follow.

Extract and Export the Keytab File

Extract the key table file, which is often referred to as the keytab, for the service principal account from the KDC and install on the machine where the Web service implementation is hosted.

For example. you can use a tool such as *kadmin.local* to extract the keytab for the service principal name, as follows:

```
>kadmin.local>ktadd -k /tmp/krb5.keytab SOAP/myhost.oracle.com
```

Export the keytab file to the machine where the Web service is hosted. The keytab is a binary file; if you ftp it, use binary mode.

Modify the krb5 Login Module to use the Keytab File

Modify the krb5 login module as described in "Configuring the SAML and Kerberos Login Modules" on page 9-15 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.oracle.com@oracle.com
- useKeyTab value=true
- storeKey value=true
- keyTab value=/scratch/myhome/krb5.keytab
- doNotPrompt value=true

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. (Authentication providers are described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.)

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.oracle.com@oracle.com*. In this example, a user with the name *SOAP/myhost.oracle.com* must exist in the identity store. Note that *@domain* should not be part of your user entry.

Creating a Ticket Cache for the Web Service Client

Perform the following steps to create a ticket cache for the Web service client:

1. Log in to the Kerberos system using the user principal you created for the client.

   ```
   >kinit fmwadmin welcome1
   ```
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@oracle.com, 1 entry found.
[1]  Service Principal:  krbtgt/oracle.com@oracle.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 system. 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.

SAML Message Protection Use Case

Assume that you have a Web service client that you want to protect with the wss11_saml_token_with_message_protection_client_policy policy, and a corresponding Web service that you want to protect with the wss11_saml_token_with_message_protection_service_policy policy.

This section steps through the procedure for using these two policies.

The following topics are described:

- "What You Need to Know" on page 9-23
  - "Requirements of the wss11_saml_token_with_message_protection_service_policy Policy" on page 9-24
  - "How Are Messages Protected Via Symmetric Keys?" on page 9-24
  - "What Keys Must Be in the Keystore?" on page 9-25
  - "Multi-Domain Use Case (Keystore Hardening)" on page 9-25
  - "When to Override the SAML Issuer" on page 9-26
- "Main Steps" on page 9-26
  - "Create a WebLogic Server User" on page 9-26
  - "Create a Java Keystore" on page 9-27
  - "Configure the Web Services Manager Keystore" on page 9-28
  - "Store the Password for the Decryption Key in the Credential Store" on page 9-29
  - "Attach the Policy to Your Web Service" on page 9-29
  - "Attach the Policy to Your Web Service Client" on page 9-29

What You Need to Know

This section describes what you need to know to configure this SAML message protection use case. The following topics are described:

- "Requirements of the wss11_saml_token_with_message_protection_service_policy Policy" on page 9-24

- "How Are Messages Protected Via Symmetric Keys?" on page 9-24
- "What Keys Must Be in the Keystore?" on page 9-25
- "Multi-Domain Use Case (Keystore Hardening)" on page 9-25
- "When to Override the SAML Issuer" on page 9-26

Requirements of the wss11_saml_token_with_message_protection_service_policy Policy

wss11_saml_token_with_message_protection_service_policy enforces message-level protection (that is, message integrity and message confidentiality), and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

Messages are protected using WS-Security's Basic 128 suite of symmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

Therefore, when you use the keytool (or other tool) to create the signature and encryption keys needed by this policy, you need to make sure you use the RSA key mechanism, the SHA-1 algorithm, and AES-128 bit encryption to satisfy the policy requirements for the key.

How Are Messages Protected Via Symmetric Keys?

This policy uses symmetric key technology. Symmetric key cryptography relies on a single, shared secret key, as follows:

1. The client creates the symmetric key, uses it to sign and encrypt the message, and shares it with the Web service in the request message.

   In order to protect the symmetric key, the symmetric key sent in the request message is encrypted using the service's certificate.

2. The Web service uses the symmetric key in the request message to verify the signature of the request message and decrypt it, and to then sign and encrypt the response message.

Consider the following process flow.

To create the request, the Oracle WSM agent does the following:

1. Generates the shared symmetric key and uses it to both sign and encrypt the request message.
2. Uses its own private key to "endorse" the signature of the request message.
3. Uses the Web service's public key to encrypt the symmetric key.
4. Sends the symmetric key along with the request to the Web service. The client sends its public key in the request so that the Web service can verify the endorsement.

When the Web service gets the request, it does the following:

1. Uses its private key to decrypt the symmetric key.
2. Uses the symmetric key to decrypt the request message and to verify its signature.

3. Uses the client's public key in the request message to verify the endorsement signature.

To send the response back to the client, the Web service does the following:

1. Uses the same client-generated symmetric key sent along with the request to sign the response message.
2. Uses the same client-generated symmetric key to encrypt the response message.

When the Oracle WSM agent receives the response message, it does the following:

1. Uses the symmetric key it generated initially to decrypt the response message.
2. Uses the symmetric key it generated initially to verify signature of the response message.

What Keys Must Be in the Keystore?

If the client and Web service are in the same domain with access to the same keystore, they can share the same private/public key pair.

That is, the client can use the private key “orakey” to endorse the signature of the request message and the public key “orakey” to encrypt the symmetric key. The Web service in turn uses the public key “orakey” to verify the endorsement, and the private key “orakey” to decrypt the symmetric key.

For demonstration purposes, this use case creates one key pair.

Multi-Domain Use Case (Keystore Hardening)

If the client and Web service are not in the same domain and do not have access to the same keystore, the client and Web service must each have a private/public key pair.

Consider the following requirements in a multiple-domain use case, as shown in Table 9–2.

Table 9–2 Multiple-Domain Use Case Requirements

Web Service Client	Web Service
Needs its own private/public key pair in the client keystore.	Needs its own private/public key pair in the service keystore.
Needs the Web service public key.	Needs the intermediary and root certificate corresponding to the client's public key in the keystore. These certificates will be used to verify the signature by generating a trusted certificate chain.
Generates symmetric key at runtime	Needs the symmetric key, but this is sent in the request message.

For the public key the client uses to encrypt the symmetric key -- that is, the public key of the Web service -- you have two approaches:

- The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension". Therefore, in this use the Web service's public key does not have to be in the client's keystore.

- As an alternative, you can specify a value for keystore.recipient.alias on the Configurations page, or override it on a per-client basis using the Security Configuration Details control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages. In this use the Web service's public key must be in the client's keystore.

When to Override the SAML Issuer

The saml.issuer.name property of the client policy identifies the issuer of the SAML token, and defaults to a value of www.oracle.com. This use case uses the www.oracle.com default.

You can optionally specify a value for saml.issuer.name on the Configurations page, or override it on a per-client basis using the Security Configuration Details control when you attach the policy.

If you do use a different SAML authority (issuer) in the policy, that issuer name must be configured in the client and included in the list of possible issuers in the SAML login module. See "Adding an Additional SAML Assertion Issuer Name" on page 9-19 for information on how to do this.

Main Steps

This section describes the steps you follow to configure the SAML message protection use case. The following topics are described:

- "Create a WebLogic Server User" on page 9-26
- "Create a Java Keystore" on page 9-27
- "Configure the Web Services Manager Keystore" on page 9-28
- "Store the Password for the Decryption Key in the Credential Store" on page 9-29
- "Attach the Policy to Your Web Service" on page 9-29
- "Attach the Policy to Your Web Service Client" on page 9-29

Create a WebLogic Server User

The user in the SAML token must already exist in the WebLogic Server identity store.

The Web service runtime extracts the SAML token from the WS-Security header and uses the name in the SAML token to validate the user against the WebLogic Server identity store.

Specifically, the SAML login module (see "Configuring the SAML and Kerberos Login Modules" on page 9-15 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) via the NameCallback to complete the perimeter authentication.

Any configured WebLogic Server authentication provider (identity asserter) that handles the JAAS NameCallback can then be invoked, including the default Authentication provider.

The WebLogic Authentication provider then simply checks whether the user exists (identity assertion mode) and, if it does, the user is asserted and a subject is established.

Create the User

You use the WebLogic Server Administration Console to add the user to the identity store, as described in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

The steps are repeated here for ease of use.

To create a user in the WebLogic Server Administration Console:

1. In the left pane select **Security Realms**.
2. On the Summary of Security Realms page select the name of the realm (for example, myrealm).
3. On the Settings for Realm Name page select **Users and Groups** and then **Users**.

   The User table displays the names of all users defined in the Authentication provider.
4. Click **New**.
5. In the Name field of the Create New User page enter the name of the user.

   User names are case sensitive and must be unique. Do not use commas, tabs or any other characters in the following comma-separated list: <>, #, |, &, ?, ( ), { }
6. (Optional) In the Description field, enter a description. The description might be the user's full name.
7. In the Provider drop-down list, select the Authentication provider for the user.

   If multiple WebLogic Authentication providers are configured in the security realm, they will appear in the list. Select which WebLogic Authentication provider's database should store information for the new user.
8. In the Password field, enter a password for the user.

   The minimum password length for a user defined in the WebLogic Authentication provider is 8 characters.
9. Re-enter the password for the user in the Confirm Password field.
10. Click **OK** to save your changes.

    The user name appears in the User table.

Create a Java Keystore

This section provides an outline of how to create and manage the Java keystore with the keytool utility. It describes how to create a keystore and load the private key and trusted CA certificates.

You can find more detailed information on the commands and arguments for the keytool utility at the following Web address: `http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html`.

Note: You specify an alias when you add an entity to the keystore using the -genkey command to generate a key pair (public and private key), or when you use the -import command to add a certificate or certificate chain to the list of trusted certificates.

Subsequent keytool commands must use this same alias to refer to the entity.

---

1. Create a new key pair and self-signed certificate.

   Use the genKey command to create the key pair (public and private key). genKey creates a new private key if one does not exist.

   The following command generates an RSA key, with RSA-SHA1 as the signature algorithm, with the alias "orakey" in the default-keystore.jks keystore. You can choose any alias name; you do not need to name your alias "orakey".

   ```
   keytool -genkey -alias orakey -keyalg "RSA" -sigalg "SHA1withRSA"-dname
   "CN=test, C=US" -keystore default-keystore.jks
   ```

   The keytool utility prompts for the needed key and keystore passwords. You need these passwords later.

2. Generate a certificate request to the certificate authority.

   Use the -certreq command to generate the request. The following commands generates a certificate request for the orakey alias.

   The CA will return a certificate or a certificate chain.

   ```
   keytool -certreq -alias orakey -sigalg "RSAwithSHA1" -file certreq_file
   -storetype jks -keystore client-default-keystore.jks
   ```

3. Replace (import) the self-signed certificate with the trusted CA certificate.

   You must replace the existing self-signed certificate with the certificate returned from the CA. To do this, use the -import command. The following command replaces the trusted CA certificate in the default-keystore.jks keystore. The keytool utility prompts for the needed password.

   ```
   keytool -import -alias orakey -file trustedcafilename -keystore
   default-keystore.jks
   ```

Configure the Web Services Manager Keystore

Perform the following steps to configure the Oracle Web Services Manager keystore:

1. In the navigator pane, expand WebLogic Domain to show the domain for which you need to configure the keystore. Select the domain.

2. Using Fusion Middleware Control, click **WebLogic Domain**, then **Security**, and then **Security Provider Configuration**.

   Click the plus sign (+) to expand the Keystore control near the bottom of the page, then click Configure.

   The Web Services Manager Keystore Configuration page is displayed, as shown in Figure 9–1.

3. If it is not already enabled, click the **Configure Keystore Management** check box.

4. Enter the path and name for the keystore that you created. By default, the keystore name is default-keystore.jks, as used in this use case. The keystore type must be JKS.

5. Enter the password for the keystore and confirm it.

6. Enter the alias and password for the signature and encryption keys.

   In this use case, orakey is the alias for both the signature and encryption keys.

   Confirm the passwords.

7. Click **OK** to submit the changes.

   Note that all fields on this page require a restart of Fusion Middleware Control to take effect.

Store the Password for the Decryption Key in the Credential Store

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

Attach the Policy to Your Web Service

Attach wss11_saml_token_with_message_protection_service_policy to your Web service as described in "Attaching a Policy to a Single Subject" on page 8-3.

Configure the policy assertion for message signing and message encryption.

The default is to sign and encrypt the entire body for the request the response. You have the option to not do this and to instead specify the specific body elements that you want to sign and encrypt. You can also additionally specify header elements that you want to sign and encrypt. Whatever you set here mush match the client policy settings.

Note: You can override keystore.sig.csf.key and keystore.enc.csf.key, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

If you do override these values, the keys for the new values must be in the keystore. That is, overriding the values does not free you from the requirement of configuring these keys in the keystores.

Attach the Policy to Your Web Service Client

Attach wss11_saml_token_with_message_protection_client_policy to your Web service client, as described in {"Attaching Policies to Web Service Clients" on page 8-9.

Configure the policy assertion for message signing, message encryption, or both.

The default is to sign and encrypt the entire body. You have the option to not do this and to instead specify the specific body elements that you want to sign and encrypt. You can also additionally specify header elements that you want to sign and encrypt. Whatever you set here must match the Web service policy settings.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10. The certificate in the WSDL is the service's public key by default, as determined by the encryption key you specified (“orakey”) when you configured the Web Services Manager keystore.

Therefore, you do not need to set or change *keystore.recipient.alias*.

You can optionally specify a value for *saml.issuer.name* on the Configurations page, or override it on a per-client basis using the Security Configuration Details control when you attach the policy. The saml.issuer.name property defaults to a value of www.oracle.com. See "When to Override the SAML Issuer" on page 9-26.

You can specify a value for *user.roles.include* on the Configurations page, or override it on a per-client basis using the Security Configuration Details control when you attach the policy.

10 Configuring Policies

This chapter discusses how to configure policies in Web services and Web service clients to achieve Quality of Service (QoS) requirements.

The predefined policies are described in Appendix B, "Predefined Policies". This Appendix is the definitive source of information for the format of the policies. Some information from the Appendix is repeated here for your convenience.

This chapter includes the following sections:

- "Determining Which Security Policies to Use" on page 10-1
- "Protecting Messages" on page 10-2
- "Authentication-Only Policies and Configuration Steps" on page 10-5
- "Message Protection-Only Policies and Configuration Steps" on page 10-10
- "Message Protection and Authentication Policies and Configuration Steps" on page 10-15
- "Authorization Policies and Configuration Steps" on page 10-44
- "WS-Addressing Policies and Configuration Steps" on page 10-52
- "MTOM Attachment Policies and Configuration Steps" on page 10-53
- "Reliable Messaging Policies and Configuration Steps" on page 10-54
- "Management Policies and Configuration Steps" on page 10-56
- "Attaching Policy Files to Web Services and Clients" on page 10-57
- "Using Client Programmatic Configuration Overrides" on page 10-58
- "Configuring Local Optimization" on page 10-66

Determining Which Security Policies to Use

Use the following series of questions to help you identify the security policies that best meet your requirements:

1. What are the **basic requirements** of your security policy? Decide if you need to only authenticate users, or if you only need message protection, or if you need both.
   a. Do you require authentication only? If yes, then go to step 2.
   b. Do you require authorization only? If yes, then see "Authorization Policies and Configuration Steps" on page 10-44
   c. Do you require authentication and authorization? If yes, then go to step 3.

   d. Do you only require message protection? If yes, then see "Message Protection-Only Policies and Configuration Steps" on page 10-10.
   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.
3. If you require **authentication and authorization**, then you need to consider the following:
   a. Review the considerations provided for authentication in step 2.
   b. Review "Authorization Policies and Configuration Steps" on page 10-44 for more information about authorization policies.
4. If you require both **authentication and message protection**, then you need to consider the following:
   a. Will message protection be handled in the transport layer? If yes, then there are four sets of policies to choose from: Username over SSL, SAML over SSL (Sender-Vouches), SAML over SSL (Token Bearer), and HTTP token over SSL.

      In one set of policies (wss_http_token_over_ssl_client_policy and wss_http_token_over_ssl_service_policy) authentication is also handled in the transport layer. For the other three polices, authentication takes place in the SOAP header.

      If you are using the WS-Security V1.0 or V1.1 standard, then both authentication and message protection occur in the SOAP header. There are five pairs of policies supporting the following tokens: username/password, SAML, and X.509 certificates.

      For more information, see "Message Protection and Authentication Policies and Configuration Steps" on page 10-15.

Protecting Messages

Message protection involves encrypting the message for message confidentiality and signing the message for message integrity. Oracle Fusion Middleware predefined policies and any policy you create using one of the message-protection assertion templates provide the options for message confidentiality, message integrity, or both.

The following steps summarizes what you must do in order to configure the clients and services for message protection:

- Attach the appropriate message protection policy to each of the clients and services.
- If you want message integrity, then the message must be signed.
- If you want message confidentiality, then the message must be encrypted.

- Add the required public and private keys to the keystores of the clients and services. This step requires you to configure the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Message Protection Basics

Message protection encompasses two concepts, **message confidentiality** and **message integrity**.

Message confidentiality involves keeping the data secret, as well as the identities of the sending and receiving parties. Confidentiality is achieved by encrypting the content of messages and obfuscating the identifies of the sending and receiving parties. The sender uses the recipient's public key to encrypt the message. Only the recipient's private key can successfully decrypt the message, ensuring that it cannot be read by third parties while in transit. The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

Message integrity is achieved by having an authority digitally sign the message. Digital signatures are used to authenticate the sender of the SOAP message and to ensure the integrity of the SOAP message (that is, to ensure that the SOAP message is not altered while in transit).

When a digital signature is applied to a SOAP message, a unique hash is produced from the message, and this hash is then encrypted with the sender's private key. When the message is received, the recipient decrypts the hash using the sender's public key.

Note: Generally, the recipient does not need to have the sender’s public key in its keystore to validate the certificate. It is sufficient to have the root certificate in the keystore to verify the certificate chain. However, if the sender's public key is not present in the message, as in the case of the Thumbprint and SerialIssuer mechanisms, the sender’s public key must be in the recipient’s keystore.

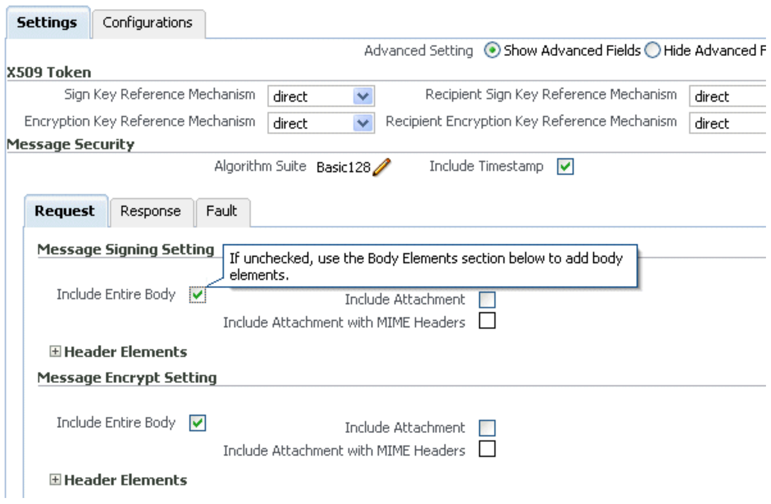
This serves to authenticate the sender, because only the sender could have encrypted the hash with the private key. It also serves to ensure that the SOAP message has not been tampered with while in transit, because the recipient can compare the hash sent with the message with a hash produced on the recipient's end.

The message-protection assertion templates and predefined policies can be used to protect request and response messages by doing the following:

- Signing messages
- Encrypting messages
- Signing *and* encrypting messages
- Decrypting messages
- Verifying signatures
- Decrypting messages *and* verifying signatures

The Fusion Middleware Control user interface for the predefined message protection policies makes it easy to specify which message parts are signed, encrypted, or both, as shown in Figure 10–1. You can require that the entire body be signed, encrypted, or both, or identity specific header and body elements.

Figure 10–1 The Signing and Encryption Portion of Message Protection Policies



Settings | Configurations

Advanced Setting [● Show Advanced Fields] ○ Hide Advanced F

X509 Token

Sign Key Reference Mechanism: direct
Recipient Sign Key Reference Mechanism: direct
Encryption Key Reference Mechanism: direct
Recipient Encryption Key Reference Mechanism: direct

Message Security

Algorithm Suite: Basic128
Include Timestamp: ✓

Request | Response | Fault

Message Signing Setting

If unchecked, use the Body Elements section below to add body elements.

Include Entire Body: ✓
Include Attachment: ☐
Include Attachment with MIME Headers: ☐

⊞ **Header Elements**

Message Encrypt Setting

Include Entire Body: ✓
Include Attachment: ☐
Include Attachment with MIME Headers: ☐

⊞ **Header Elements**

Security SwA Attachments

Packaging as attachments in SOAP messages has become a norm in the Web services area 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 Attachment* signs the attachment but not the MIME header corresponding to that. *Include Attachment with MIME Headers* signs the attachments as well as the MIME headers.

Which Policies Offer Message Protection?

The following policies offer message protection. The subsequent sections for each of these policies later in this chapter describe how each policy implements message protection.

- oracle/wss10_message_protection_client_policy
- oracle/wss10_message_protection_service_policy
- oracle/wss10_username_id_propagation_with_msg_protection_client_policy
- oracle/wss10_username_id_propagation_with_msg_protection_service_policy
- oracle/wss10_username_token_with_message_protection_client_policy
- oracle/wss10_username_token_with_message_protection_service_policy
- oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy
- oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy
- oracle/wss10_x509_token_with_message_protection_client_policy
- oracle/wss10_x509_token_with_message_protection_service_policy
- oracle/wss10_saml_token_with_message_protection_client_policy
- oracle/wss10_saml_token_with_message_protection_service_policy

- oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy
- oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy
- oracle/wss11_message_protection_client_policy
- oracle/wss11_message_protection_service_policy
- oracle/wss11_kerberos_token_with_message_protection_client_policy
- oracle/wss11_kerberos_token_with_message_protection_service_policy
- oracle/wss11_saml_token_with_message_protection_client_policy
- oracle/wss11_saml_token_with_message_protection_service_policy
- oracle/wss11_username_token_with_message_protection_client_policy
- oracle/wss11_username_token_with_message_protection_service_policy
- oracle/wss11_x509_token_with_message_protection_client_policy
- oracle/wss11_x509_token_with_message_protection_service_policy

Both the WS-Security 1.0 and WS-Security 1.1 standards are supported. Use the assertion template or predefined policy that supports the standard which both the Web service and client share in common. If you are starting anew, use the WS-Security 1.1 standard because it provides more options and requires less PKI deployment.

The assertion templates support partial signing and encryption as well as full signing and encryption of the message body. For those assertion templates or predefined policies that provide SOAP message protection, the default behavior is to protect the entire SOAP message body by signing and encrypting the entire SOAP body. You can configure the assertions and policies to protect selected elements, if you wish.

Authentication-Only Policies and Configuration Steps

Table B–1 in Appendix B, "Predefined Policies" summarizes the security policies that enforce authentication only, and indicates whether the token is inserted at the transport layer or SOAP header.

This section lists the authentication-only predefined policies, indicates the type of Web service to which they apply, and provides a link to the configuration steps you must perform to use them.

oracle/wss_http_token_client_policy

The oracle/wss_http_token_client_policy policy includes credentials in the HTTP header for outbound client requests. It is the analogous client policy to the oracle/wss_http_token_service_policy service endpoint policy.

This policy contains the following policy assertion: oracle/wss_http_token_client_template. See "oracle/wss_http_token_client_template" on page 2 for more information about the assertion.

Settings You Can Change

See Table C–2.

Properties You Can Configure

See Table C–3.

How to Set Up the Web Service Client

You can specify a value for *csf-key* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

The value signifies a key that maps to a username/password. See "Configuring the Credential Store Provider" on page 9-12 for information on how to add the key to the credential store.

If you do not set the **Require Mutual Authentication** control, SSL is not involved. If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 9-7.

How to Set Up the Web Service Client at Design Time

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

The client must pass the credentials in the HTTP header.

If you do not set the **Require Mutual Authentication** control, SSL is not involved. If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 9-7.

oracle/wss_http_token_service_policy

The wss_http_token_service_policy uses the credentials in the HTTP header to authenticate users.

This policy contains the following policy assertion: oracle/wss_http_token_service_template. See "oracle/wss_http_token_service_template" on page 4 for more information about the assertion.

Settings You Can Change

See Table C–2.

Properties You Can Configure

See Table C–4.

How to Set Up WebLogic Server

The Web service must authenticate the supplied username and password credentials against the configured authentication source.

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

For mutual SSL authentication, you must configure WebLogic Server. See "Configuring SSL on WebLogic Server (Two-Way)" on page 9-5.

oracle/wss_username_token_client_policy

This policy includes credentials in the WS-Security UsernameToken header for all outbound SOAP request messages. A plain text mechanism is supported, in addition to a password not being required. It is the analogous client policy to the oracle/wss_username_token_service_policy service endpoint policy.

This policy contains the following policy assertion: oracle/wss_username_token_client_template. See "oracle/wss_username_token_client_template" on page C-5 for more information about the assertion.

Settings You Can Change

See Table C–5.

Properties You Can Configure

See Table C–6.

How to Set Up the Web Service Client

You can specify a value for *csf-key* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

The value signifies a key that maps to a username/password. See "Configuring the Credential Store Provider" on page 9-12 for information on how to add the key to the credential store.

If you specify a password type of None on the **Settings** page, you do not need to include a password in the key.

How to Set Up the Web Service Client At Design Time

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

The client must include a WS-Security UsernameToken element (<wsse:UsernameToken/>) in the SOAP request message. The client provides a username and password for authentication.

oracle/wss_username_token_service_policy

This policy uses the credentials in the UsernameToken WS-Security SOAP header to authenticate users. The plain text mechanism is supported.

This policy contains the following policy assertion: oracle/wss_username_token_service_template. See "oracle/wss_username_token_service_template" on page C-7 for more information about the assertion.

Settings You Can Change

See Table C–5.

Properties You Can Configure

See Table C–7.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

oracle/wss10_saml_token_client_policy

This policy includes SAML tokens in outbound SOAP request messages.

This policy contains the following policy assertion: oracle/wss10_saml_token_client_template. See "oracle/wss10_saml_token_client_template" on page 8 for more information about the assertion.

Settings You Can Change

See Table C–8.

Properties You Can Configure

See Table C–9.

How to Set Up the Web Service Client

See "Configuring SAML" on page 9-16.

You can specify a value for *saml.issuer.name* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The *saml.issuer.name* property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 9-19 for additional considerations.

How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 9-17.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

Include a WS-Security Header Element (<saml:Assertion>) that inserts a SAML token in the outbound SOAP message. The confirmation type is always *sender-vouches*.

oracle/wss10_saml_token_service_policy

This policy authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header.

This policy contains the following policy assertion: oracle/wss10_saml_token_service_template. See "oracle/wss10_saml_token_service_template" on page C-9 for more information about the assertion.

Settings You Can Change

See Table C–8.

Properties You Can Configure

See Table C–10.

Configure the Login Module

Configure the *saml.loginmodule* login module. See "Configuring the SAML and Kerberos Login Modules" on page 9-15 for more information.

How to Set Up Oracle Platform Security Services (OPPS)

See "Configuring SAML" on page 9-16.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

The SAML login module extracts the username from the verified token and passes it (via the *NameCallback*) to the provider.

oracle/wss11_kerberos_token_client_policy

This policy includes a Kerberos token in the WS-Security header in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

Service principal names (SPN) are a key component in Kerberos authentication. SPNs are unique identifiers for services running on servers. Every service that uses Kerberos authentication needs to have an SPN set for it so that clients can identify the service on the network. If an SPN is not set for a service, clients have no way of locating that service and Kerberos authentication is not possible.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_client_template. See "oracle/wss11_kerberos_token_with_message_protection_client_template" on page C-37 for more information about the assertion.

Settings You Can Change

See Table C–42.

Properties You Can Configure

See Table C–43.

How to Set Up the Web Service Client

See "Using Kerberos Tokens" on page 9-19.

The Web service client that is enforcing Kerberos client side policies needs to know the service principal name of the service it is trying to access. You can specify a value for *service.principal.name* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The default value (place holder) is *HOST/localhost@oracle.com*.

How to Set Up the Web Service Client at Design Time

See "Using Kerberos Tokens" on page 9-19.

You must set the service principal name. The service principal name specifies the name of the service principal for which the client requests a ticket from the KDC.

If the Kerberos authentication is successful, then send the obtained Kerberos ticket and authenticator to the Web service enclosed in a BinarySecurityToken element in the SOAP Security header.

oracle/wss11_kerberos_token_service_policy

This policy is enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

Service principal names (SPN) are a key component in Kerberos authentication. SPNs are unique identifiers for services running on servers. Every service that uses Kerberos authentication needs to have an SPN set for it so that clients can identify the service on the network. If an SPN is not set for a service, clients have no way of locating that service and Kerberos authentication is not possible.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_service_template. See "oracle/wss11_kerberos_token_with_message_protection_service_template" on page C-39 for more information about the assertion.

Settings You Can Change

See Table C–42.

Properties You Can Configure

See Table C–44.

Configure the Login Module

Configure the *krb5.loginmodule* login module. See "Configuring the SAML and Kerberos Login Modules" on page 9-15 for more information.

How to Configure WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

Message Protection-Only Policies and Configuration Steps

See "Protecting Messages" on page 10-2 for a description of how the predefined policies implement message protection.

Table B–2 summarizes the policies that enforce only message protection, and indicates whether the policy is enforced at the transport layer or SOAP header.

Message protection-only policies do not authenticate or authorize the requester.

There may be either one or two Security policies attached to a policy subject. A Security policy can contain an assertion that belongs to the authentication or message protection (as in this case) subtype categories, or a single assertion that belongs to both subtype categories. You can then use an assertion that belongs to the authorization subtype to authorize the requester.

oracle/wss10_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss10_message_protection_client_template. See "oracle/wss10_message_protection_client_policy" on page B-4 for more information about the assertion.

Settings You Can Change

See Table C–15.

Properties You Can Configure

See Table C–16.

How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

As an alternative, you can specify a value for *keystore.recipient.alias* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The *keystore.recipient.alias* specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for *keystore.sig.csf.key* and *keystore.enc.csf.key* on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

How to Set Up the Web Service Client at Design Time

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 10–1 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 10–1 WS-Security 1.0 Message Integrity of SOAP Message

```
<dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
 <dsig:SignedInfo>
  <dsig:CanonicalizationMethod
   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
  <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <dsig:Reference URI="#Timestamp-...">
     <dsig:Transforms>
       <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
     </dsig:Transforms>
     <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
     <dsig:DigestValue>...</dsig:DigestValue>
  </dsig:Reference>
  <dsig:Reference URI="#Body-...">
     <dsig:Transforms>
         <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
     </dsig:Transforms>
     <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
     <dsig:DigestValue>...</dsig:DigestValue>
  </dsig:Reference>
  <dsig:Reference URI="#KeyInfo-...">
   <dsig:Transforms>
     <dsig:Transform
Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-se
```

```
curity-1.0#STR-Transform">
       <TransformationParameters
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1
.0.xsd">
       <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
xmlns="http://www.w3.org/2000/09/xmldsig#"/>
       </TransformationParameters>
     </dsig:Transform>
   </dsig:Transforms>
   <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
   <dsig:DigestValue>...</dsig:DigestValue>
  </dsig:Reference>
 </dsig:SignedInfo>
 <dsig:SignatureValue>....</dsig:SignatureValue>
 <dsig:KeyInfo Id="KeyInfo-...">
     <wsse:SecurityTokenReference
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1
.0.xsd">
      <wsse:KeyIdentifier
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-prof
ile-1.0#X509SubjectKeyIdentifier"
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message
-security-1.0#Base64Binary">
...</wsse:KeyIdentifier>
     </wsse:SecurityTokenReference>
 </dsig:KeyInfo>
</dsig:Signature>
```

Example 10–2 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element is encrypted.

Example 10–2 WS-Security 1.0 Message Confidentiality of SOAP Message

```
<env:Body
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-util
ity-1.0.xsd" wsu:Id="Body-JA9fsCRnqbFJ0ocBAMKb7g22">
 <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#Content" Id="...">
  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
  <xenc:CipherData>
      <xenc:CipherValue>...</xenc:CipherValue>
  </xenc:CipherData>
 </xenc:EncryptedData>
</env:Body>
```

oracle/wss10_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

The messages are protected using WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. This policy does not authenticate or authorize the requester.

This policy contains the following policy assertion: oracle/wss10_message_protection_service_template. See "oracle/wss10_message_protection_service_template" on page C-13 for more information about the assertion.

Settings You Can Change

See Table C–15.

Properties You Can Configure

See Table C–17. You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Store the trusted certificate that corresponds to the client’s private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

oracle/wss11_message_protection_client_policy

This policy provides message integrity and confidentiality for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following policy assertion: oracle/wss11_message_protection_client_template. See "oracle/wss11_message_protection_client_template" on page C-14 for more information about the assertion.

Settings You Can Change

See Table C–18.

Properties You Can Configure

See Table C–19.

How to Configure the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

As an alternative, you can specify a value for *keystore.recipient.alias* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for *keystore.enc.csf.key* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

How to Configure the Web Service Client at Design Time

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

This policy uses symmetric key technology, which is an encryption method that uses the same shared key to encrypt and decrypt data. The symmetric key is used to sign the message.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 10–3 is an example of the typical structure of encryption elements included in the Security header in conformance with the WS-Security 1.1 standards. In this example, the body element is encrypted.

Example 10–3   WS-Security 1.1 Message Confidentiality of SOAP Message

```
<xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="EK-...">
<xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
<dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" />
</xenc:EncryptionMethod>
<dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
<wsse:SecurityTokenReference
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1
.0.xsd">
<wsse:KeyIdentifier
ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#Thum
bprintSHA1"
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message
-security-1.0#Base64Binary">...</wsse:KeyIdentifier>
</wsse:SecurityTokenReference>
</dsig:KeyInfo>
<xenc:CipherData>
<xenc:CipherValue>...</xenc:CipherValue>
</xenc:CipherData>
<xenc:ReferenceList>
<xenc:DataReference URI="#_..." />
</xenc:ReferenceList>
</xenc:EncryptedKey>
<env:Body
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-util
ity-1.0.xsd" wsu:Id="Body-...">
  <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#Content" Id="...">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"
/>
    <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
      <wsse:SecurityTokenReference
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-sec
ext-1.0.xsd"
```

```
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1
.0.xsd">
         <wsse:Reference URI="#EK-..."
ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#Encr
yptedKey" />
      </wsse:SecurityTokenReference>
    </dsig:KeyInfo>
    <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</env:Body>
```

oracle/wss11_message_protection_service_policy

This policy enforces message integrity and confidentiality for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following policy assertion: oracle/wss11_message_protection_service_template. See "oracle/wss11_message_protection_service_template" on page C-15 for more information about the assertion.

Settings You Can Change

See Table C–18.

Properties You Can Configure

See Table C–20. You also have the option to override the *keystore.enc.csf.key* server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.enc.csf.key* server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

Message Protection and Authentication Policies and Configuration Steps

Table B–3 summarizes the policies that enforce both message protection and authentication, and indicates whether the policy is enforced at the transport layer or SOAP header. These polices are described in the sections that follow.

See "Protecting Messages" on page 10-2 for a description of how the predefined policies implement message protection.

Configuring a Policy With an OR Group

The oracle/wss11_saml_or_username_token_with_message_protection_service_policy and oracle/wss_saml_or_username_token_over_ssl_service_policy policies contain assertions as an OR group--meaning that either type of assertion can be enforced by a client.

In addition, you can add an OR group to the policy of your choice, as described in "Adding an OR Group to a Policy" on page 7-8.

The oracle/wss11_saml_or_username_token_with_message_protection_service_policy policy contains the following assertions:

- oracle/wss11_saml_token_with_message_protection_service_template. See "oracle/wss11_saml_token_with_message_protection_service_policy" on page 10-40 for information about configuring the policy.
- oracle/wss11_username_token_with_message_protection_service_template. See "oracle/wss11_username_token_with_message_protection_service_policy" on page 10-42 for information about configuring the policy.

The oracle/wss_saml_or_username_token_over_ssl_service_policy policy contains the following assertions:

- oracle/wss_saml_token_over_ssl_service_template. See "oracle/wss_saml_token_over_ssl_service_policy" on page 10-20 for information about configuring the policy.
- oracle/wss_username_token_over_ssl_service_template. See "oracle/wss_username_token_over_ssl_service_policy" on page 10-21 for information about configuring the policy.

oracle/wss_http_token_over_ssl_client_policy

This policy includes credentials in the HTTP header for outbound client requests.

This policy also verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be applied to any HTTP-based endpoint.

Note: Currently only HTTP basic authentication is supported.

This policy contains the following policy assertion: oracle/wss_http_token_over_ssl_client_template. See "oracle/wss_http_token_over_ssl_client_template" on page C-17 for more information about the assertion.

Setting You Can Change

See Table C–22.

Properties You Can Configure

See Table C–23.

How to Set Up the Web Services Client

You can specify a value for *csf-key* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

The value signifies a key that maps to a username/password. See "Configuring the Credential Store Provider" on page 9-12 for information on how to add the key to the credential store.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved. See "Configuring SSL for a Web Service Client" on page 9-6.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 9-7.

How to Set Up the Web Service Client at Design Time

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

The client must pass the credentials in the HTTP header.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved. See "Configuring SSL for a Web Service Client" on page 9-6.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 9-7.

oracle/wss_http_token_over_ssl_service_policy

This policy extracts the credentials in the HTTP header and authenticates users.

This policy verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be applied to any HTTP-based endpoint.

Note: Currently only HTTP basic authentication is supported.

This policy contains the following policy assertion: oracle/wss_http_token_over_ssl_service_template. See "oracle/wss_http_token_over_ssl_service_template" on page C-19 for more information about the assertion.

Settings You Can Change

See Table C–22.

Properties You Can Configure

See Table C–24.

How to Set Up WebLogic Server

Configure SSL, as described in "Configuring SSL on WebLogic Server (One-Way)" on page 9-4, or as in "Configuring SSL on WebLogic Server (Two-Way)" on page 9-5 if **Allow Mutual Authentication** is checked.

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

oracle/wss_saml_token_bearer_over_ssl_client_policy

This policy includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method *Bearer* is created automatically.

This policy contains the following policy assertion: oracle/wss_saml_token_bearer_over_ssl_client_template. See "oracle/wss_saml_token_bearer_over_ssl_client_template" on page C-20 for more information about the assertion.

Settings You Can Change

See Table C–25

Properties You Can Configure

None.

How to Set Up the Web Service Client

See "How to Configure SAML Web Service Client at Design Time" on page 9-17.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 9-6.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 9-7.

How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 9-17.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 9-6.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 9-7.

oracle/wss_saml_token_bearer_over_ssl_service_policy

This policy authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.

This policy contains the following policy assertion: oracle/wss_saml_token_bearer_over_ssl_service_template. See "oracle/wss_saml_token_bearer_over_ssl_service_template" on page C-21 for more information about the assertion.

Settings You Can Change

See Table C–25.

Properties You Can Configure

None.

Configure the Login Module

Configure the *saml.loginmodule* login module. See "Configuring the SAML and Kerberos Login Modules" on page 9-15 for more information.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 9-16.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

The SAML login module extracts the username from the verified token and passes it (via the *NameCallback*) to the Authentication provider.

To configure SSL, see "Configuring SSL on WebLogic Server (One-Way)" on page 9-4, or "Configuring SSL on WebLogic Server (Two-Way)" on page 9-5 if **Require Mutual Authentication** is checked.

oracle/wss_saml_token_over_ssl_client_policy

This policy enables the authentication of credentials provided via a SAML token within WS-Security SOAP header.

This policy contains the following policy assertion: oracle/wss_saml_token_over_ssl_client_template. See "oracle/wss_saml_token_over_ssl_client_template" on page C-21 for more information about the assertion.

Settings You Can Change

See Table C–26.

Properties You Can Configure

None.

How to Set Up the Web Service Client

See "How to Configure SAML Web Service Client at Design Time" on page 9-17.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 9-6.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 9-7.

How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 9-17.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 9-6.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 9-7.

oracle/wss_saml_token_over_ssl_service_policy

This policy enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header.

This policy contains the following policy assertion: oracle/wss_saml_token_over_ssl_service_template. See "oracle/wss_saml_token_over_ssl_service_template" on page C-21 for more information about the assertion.

Settings You Can Change

See Table C–26

Properties You Can Configure

None.

Configure the Login Module.

Configure the *saml.loginmodule* login module. See "Configuring the SAML and Kerberos Login Modules" on page 9-15 for more information.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 9-16.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

The SAML login module extracts the username from the verified token and passes it (via the *NameCallback*) to the Authentication provider.

To configure SSL, see "Configuring SSL on WebLogic Server (One-Way)" on page 9-4, or "Configuring SSL on WebLogic Server (Two-Way)" on page 9-5 if **Require Mutual Authentication** is checked.

oracle/wss_username_token_over_ssl_client_policy

This policy includes credentials in the WS-Security UsernameToken header in outbound SOAP request messages. The plain text mechanism is supported. The policy also uses SSL for achieving transport layer security.

This policy contains the following policy assertion: oracle/wss_username_token_over_ssl_client_template. See "oracle/wss_username_token_over_ssl_client_template" on page C-21 for more information about the assertion.

Settings You Can Change

See Table C–27.

Properties You Can Configure

See Table C–28.

How to Set Up the Web Service Client

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved, as described in "Configuring SSL for a Web Service Client" on page 9-6.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 9-7.

You can specify a value for *csf-key* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

The value signifies a key that maps to a username/password. See "Configuring the Credential Store Provider" on page 9-12 for information on how to add the key to the credential store.

If you specify a password type of None on the **Settings** page, you do not need to include a password in the key.

How to Set Up the Web Service Client at Design Time

The client must include a WS-Security UsernameToken element (<wsse:UsernameToken/>) in the SOAP request message. The client provides a username and password for authentication.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

If you do not set the **Require Mutual Authentication** control, one-way SSL is involved. See "Configuring SSL for a Web Service Client" on page 9-6.

If you do set the **Require Mutual Authentication** control, the client must supply credentials, and expect credentials back from the Web service. See "Configuring Two-Way SSL for a Web Service Client" on page 9-7.

oracle/wss_username_token_over_ssl_service_policy

This policy uses the credentials in the UsernameToken WS-Security SOAP header to authenticate users. The plain text mechanism is supported.

This policy contains the following policy assertion: oracle/wss_username_token_over_ssl_service_template. See "oracle/wss_username_token_over_ssl_service_template" on page C-23 for more information about the assertion.

Settings You Can Change

See Table C–27.

Properties You Can Configure

See Table C–29.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

The username and password must exist and be valid.

To configure SSL, see "Configuring SSL on WebLogic Server (One-Way)" on page 9-4, or "Configuring SSL on WebLogic Server (Two-Way)" on page 9-5 if **Require Mutual Authentication** is checked.

oracle/wss10_saml_hok_token_with_message_protection_client_policy

This policy provides message-level protection and SAML holder of key based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss10_saml_hok_with_message_integrity_client_template. See "oracle/wss10_saml_hok_with_message_protection_service_template" on page C-27 for more information about the assertion.

Settings You Can Change

See Table C–30.

Properties You Can Configure

See Table C–31.

How to Set Up the Web Service Client

See "How to Configure SAML Web Service Client at Design Time" on page 9-17.

Override the *saml.assertion.filename* property to point to the file that has the holder-of-key assertion.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

You can specify a value for *saml.issuer.name* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The *saml.issuer.name* property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 9-19 for additional considerations.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

As an alternative, you can specify a value for *keystore.recipient.alias* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for *keystore.sig.csf.key* and *keystore.enc.csf.key* on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 9-17.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

Override the saml.assertion.filename property to point to the file that has the holder-of-key assertion. See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 10–1 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 10–2 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.

oracle/wss10_saml_hok_token_with_message_protection_service_policy

This policy enforces message-level protection and SAML holder of key based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss10_saml_hok_with_message_integrity_service_template. See "oracle/wss10_saml_hok_with_message_protection_service_template" on page C-27 for more information about the assertion.

Configure the Login Module

Configure the *saml.loginmodule* login module. See "Configuring the SAML and Kerberos Login Modules" on page 9-15 for more information.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

The SAML login module extracts the username from the verified token and passes it (via the *NameCallback*) to the Authentication provider.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 9-16.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Note: A CertificateExpiredException is returned if an expired certificate is present in the keystore, regardless of whether this certificate is being referenced. To resolve this exception, remove the expired certificate from the keystore.

Store the trusted certificate of the SAML authority in the keystore.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

oracle/wss10_saml_token_with_message_integrity_client_policy

This policy provides message-level integrity and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_integrity_client_template. See "oracle/wss10_saml_token_with_message_protection_client_template" on page C-28 for more information about the assertion.

Settings You Can Change

See Table C–33.

Properties You Can Configure

See Table C–34.

How to Set Up the Web Service Client

See "Configuring SAML" on page 9-16.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

You can specify a value for *saml.issuer.name* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The *saml.issuer.name* property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 9-19 for additional considerations.

You can specify a value for *user.roles.include* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for *keystore.sig.csf.key* and *keystore.enc.csf.key* on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 9-17.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

Include a WS-Security Header Element (<saml:Assertion>) that inserts a SAML token in the outbound SOAP message. The confirmation type is always *sender-vouches*.

Example 10–1 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 10–2 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.

oracle/wss10_saml_token_with_message_integrity_service_policy

This policy enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_integrity_service_template. See "oracle/wss10_saml_token_with_message_protection_service_template" on page C-30 for more information about the assertion.

Settings You Can Change

See Table C–33.

Properties You Can Configure

See Table C–35.

You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

Configure the Login Module

Configure the *saml.loginmodule* login module. See "Configuring the SAML and Kerberos Login Modules" on page 9-15 for more information.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 9-16.

You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

The SAML login module extracts the username from the verified token and passes it (via the *NameCallback*) to the Authentication provider.

oracle/wss10_saml_token_with_message_protection_client_policy

This policy provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_client_template. See "oracle/wss10_saml_token_with_message_protection_client_template" on page C-28 for more information about the assertion.

Settings You Can Change

See Table C–33.

Properties You Can Configure

See Table C–34.

How to Set Up the Web Service Client

See "Configuring SAML" on page 9-16.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

As an alternative, you can specify a value for *keystore.recipient.alias* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for *keystore.sig.csf.key* and *keystore.enc.csf.key* on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for *saml.issuer.name* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The *saml.issuer.name* property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 9-19 for additional considerations.

You can specify a value for *user.roles.include* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 9-17.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 10–1 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 10–2 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.

oracle/wss10_saml_token_with_message_protection_service_policy

This policy enforces message-level protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_service_template. See "oracle/wss10_saml_token_with_message_protection_service_template" on page C-30 for more information about the assertion.

Settings You Can Change

See Table C–33.

Properties You Can Configure

See Table C–35.

You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

Configure the Login Module

Configure the *saml.loginmodule* login module. See "Configuring the SAML and Kerberos Login Modules" on page 9-15 for more information.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

The SAML login module extracts the username from the verified token and passes it (via the *NameCallback*) to the Authentication provider.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 9-16.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Store the trusted certificate that corresponds to the client’s private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy

This policy provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

This policy uses the Subject Key Identifier (ski) reference mechanism for the encryption key in the request, and for both the signature and encryption keys in the response.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_client_template. See "oracle/wss10_saml_token_with_message_protection_client_template" on page C-28 for more information about the assertion.

Settings You Can Change

See Table C–33.

Properties You Can Configure

See Table C–34.

How to Set Up the Web Service Client

See "Configuring SAML" on page 9-16.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

As an alternative, you can specify a value for *keystore.recipient.alias* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for *keystore.sig.csf.key* and *keystore.enc.csf.key* on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for *saml.issuer.name* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The *saml.issuer.name* property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 9-19 for additional considerations.

You can specify a value for *user.roles.include* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 9-17.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 10–1 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 10–2 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.

oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy

This policy enforces message-level protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses the Subject Key Identifier (ski) reference mechanism for the encryption key in the request, and for both the signature and encryption keys in the response.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_service_template. See "oracle/wss10_saml_token_with_message_protection_service_template" on page C-30 for more information about the assertion.

Settings You Can Change

See Table C–33.

Properties You Can Configure

See Table C–35.

You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

Configure the Login Module

Configure the *saml.loginmodule* login module. See "Configuring the SAML and Kerberos Login Modules" on page 9-15 for more information.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

The SAML login module extracts the username from the verified token and passes it (via the *NameCallback*) to the Authentication provider.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 9-16.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore. When using the ski reference mechanism, use OpenSSL or another such utility to create the certificate.

Store the trusted certificate that corresponds to the client’s private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

oracle/wss10_username_id_propagation_with_msg_protection_client_policy

This policy provides message-level protection (that is, integrity and confidentiality) and identity propagation for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss10_username_id_propagation_with_msg_protection_client_template. See "oracle/wss10_username_token_with_message_protection_client_template" on page C-31 for more information about the assertion.

Settings You Can Change

See Table C–36.

Properties You Can Configure

See Table C–37.

How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

As an alternative, you can specify a value for *keystore.recipient.alias* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for *keystore.sig.csf.key* and *keystore.enc.csf.key* on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

How to Set Up the Web Service Client at Design Time

The client must include a WS-Security UsernameToken element (<wsse:UsernameToken/>) in the SOAP request message. The client provides a username and password for authentication.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8. The policy specifically requires that the client’s and Web service’s respective keystores already contain digital certificates containing each other’s public key.

Configure the policy assertion for message signing, message encryption, or both.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

Example 10–1 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 10–2 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.

oracle/wss10_username_id_propagation_with_msg_protection_service_policy

This policy enforces message level protection (that is, integrity and confidentiality) and identity propagation for inbound SOAP requests using mechanisms described in WS-Security 1.0.

This policy contains the following policy assertion: oracle/wss10_username_id_propagation_with_msg_protection_service_template. See "oracle/wss10_username_token_with_message_protection_service_template" on page C-34 for more information about the assertion.

Settings You Can Change

See Table C–37.

Properties You Can Configure

See Table C–39. You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

The SAML login module extracts the username from the verified token and passes it (via the *NameCallback*) to the Authentication provider.

How to Set Up Oracle Platform Security Services (OPSS

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Store the trusted certificate that corresponds to the client’s private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

oracle/wss10_username_token_with_message_protection_client_policy

This policy provides message-level protection (message integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_client_template. See "oracle/wss10_username_token_with_message_protection_client_template" on page C-31 for more information about the assertion.

Settings You Can Change

See Table C–36.

Properties You Can Configure

See Table C–37.

How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

As an alternative, you can specify a value for *keystore.recipient.alias* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for *keystore.sig.csf.key* and *keystore.enc.csf.key* on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for *csf-key* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

The value signifies a key that maps to a username/password. See "Configuring the Credential Store Provider" on page 9-12 for information on how to add the key to the credential store.

How to Set Up the Web Service Client at Design Time

Configure the policy assertion for message signing, message encryption, or both.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8. The policy specifically requires that the client’s and Web service’s respective keystores already contain digital certificates containing each other’s public key.

Example 10–1 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 10–2 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.

oracle/wss10_username_token_with_message_protection_service_policy

This policy enforces message-level protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_service_template. See "oracle/wss10_username_token_with_message_protection_service_template" on page C-34 for more information about the assertion.

Settings You Can Change

See Table C–36.

Properties You Can Configure

See Table C–38. You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy

This policy provides message-level protection (message integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses the Subject Key Identifier (ski) reference mechanism for the encryption key in the request, and for both the signature and encryption keys in the response.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_client_template. See "oracle/wss10_username_token_with_message_protection_client_template" on page C-31 for more information about the assertion.

Settings You Can Change

See Table C–36.

Properties You Can Configure

See Table C–37.

How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

As an alternative, you can specify a value for *keystore.recipient.alias* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for *keystore.sig.csf.key* and *keystore.enc.csf.key* on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for *csf-key* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

The value signifies a key that maps to a username/password. See "Configuring the Credential Store Provider" on page 9-12 for information on how to add the key to the credential store.

How to Set Up the Web Service Client at Design Time

Configure the policy assertion for message signing, message encryption, or both.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8. The policy specifically requires that the client’s and Web service’s respective keystores already contain digital certificates containing each other’s public key.

Example 10–1 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 10–2 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.

oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy

This policy enforces message-level protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses the Subject Key Identifier (ski) reference mechanism for the encryption key in the request, and for both the signature and encryption keys in the response.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_service_template. See "oracle/wss10_username_token_with_message_protection_service_template" on page C-34 for more information about the assertion.

Settings You Can Change

See Table C–36.

Properties You Can Configure

See Table C–38. You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore. When using the ski reference mechanism, use OpenSSL or another such utility to create the certificate.

Store the trusted certificate that corresponds to the client’s private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

oracle/wss10_x509_token_with_message_protection_client_policy

This policy provides message-level protection and certificate credential population for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss10_x509_token_with_message_protection_client_template. See "oracle/wss10_x509_token_with_message_protection_client_template" on page C-35 for more information about the assertion.

Settings You Can Change

See Table C–39.

Properties You Can Configure

See Table C–40.

How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

As an alternative, you can specify a value for *keystore.recipient.alias* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for *keystore.sig.csf.key* and *keystore.enc.csf.key* on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

How to Set Up the Web Service Client at Design Time

The Web service client needs to provide valid X.509 authentication credentials in the SOAP message through the WS-Security binary security token.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 10–1 shows the typical structure of a signature included in the Security header in conformance with the WS-Security 1.0 standards. In this example, the body element of the SOAP message is signed.

Example 10–2 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.

oracle/wss10_x509_token_with_message_protection_service_policy

This policy enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss10_x509_token_with_message_protection_service_template. See "oracle/wss10_x509_token_with_message_protection_service_template" on page C-37 for more information about the assertion.

Settings You Can Change

See Table C–39.

Attributes You Can Configure

See Table C–41. You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.sig.csf.key* and *keystore.enc.csf.key* server-side configuration properties, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up WebLogic Server

You need to configure an Authentication provider, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14, and make sure that you provide the X.509 callback information for this provider.

oracle/wss11_kerberos_token_with_message_protection_client_policy

This policy includes a Kerberos token in the WS-Security header, and uses Kerberos keys to guarantee message integrity and confidentiality, in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_with_message_protection_client_template. See "oracle/wss11_kerberos_token_with_message_protection_client_template" on page C-37 for more information about the assertion.

Settings You Can Change

See Table C–42.

Properties You Can Configure

See Table C–43.

How to Set up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

You can specify a value for *keystore.sig.csf.key* and *keystore.enc.csf.key* on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

Also see "Using Kerberos Tokens" on page 9-19.

How to Set Up the Web Service Client at Design Time

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8.

Also see "Using Kerberos Tokens" on page 9-19.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 10–3 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.

oracle/wss11_kerberos_token_with_message_protection_service_policy

This policy is enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_with_message_protection_service_template. See "oracle/wss11_kerberos_token_with_message_protection_service_template" on page C-39 for more information about the assertion.

Settings You Can Change

See Table C–42.

Properties You Can Configure

See Table C–44. You also have the option to override the *keystore.enc.csf.key* server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

Configure the Login Module

Configure the *krb5.loginmodule* login module. See "Configuring the SAML and Kerberos Login Modules" on page 9-15.

How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Store the trusted certificate that corresponds to the client’s private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.enc.csf.key* server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

Configure Kerberos, as described in "Using Kerberos Tokens" on page 9-19.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

oracle/wss11_saml_token_with_message_protection_client_policy

This policy enables message level protection and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1.

This policy contains the following policy assertion: oracle/wss11_saml_token_with_message_protection_client_template. See "oracle/wss11_saml_token_with_message_protection_client_template" on page C-40 for more information about the assertion.

Settings You Can Change

See Table C–45.

Properties You Can Configure

See Table C–46.

How to Set Up the Web Service Client

See "How to Configure SAML Web Service Client at Design Time" on page 9-17.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

As an alternative, you can specify a value for *keystore.recipient.alias* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for *keystore.sig.csf.key* and *keystore.enc.csf.key* on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

You can specify a value for *saml.issuer.name* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The *saml.issuer.name* property defaults to a value of www.oracle.com. See "Adding an Additional SAML Assertion Issuer Name" on page 9-19 for additional considerations.

You can specify a value for *user.roles.include* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

How to Set Up the Web Service Client at Design Time

See "How to Configure SAML Web Service Client at Design Time" on page 9-17.

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 10–3 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.

oracle/wss11_saml_token_with_message_protection_service_policy

This policy enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy contains the following policy assertion: oracle/wss11_saml_token_with_message_protection_service_template. See "oracle/wss11_saml_token_with_message_protection_service_template" on page C-42 for more information about the assertion.

Settings You Can Change

See Table C–45.

Properties You Can Configure

See Table C–46.

You also have the option to override the *keystore.enc.csf.key* server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

Configure the Login Module

Configure the *saml.loginmodule* login module. See "Configuring the SAML and Kerberos Login Modules" on page 9-15.

How to Set Up Oracle Platform Security Services (OPSS)

See "Configuring SAML" on page 9-16.

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.enc.csf.key* server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

The SAML login module extracts the username from the verified token and passes it (via the *NameCallback*) to the Authentication provider.

oracle/wss11_username_token_with_message_protection_client_policy

This policy provides message-level protection and authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following policy assertion: oracle/wss11_username_token_with_message_protection_client_template. See "oracle/wss11_username_token_with_message_protection_client_template" on page C-43 for more information about the assertion.

Settings You Can Change

See Table C–48.

Properties You Can Configure

See Table C–49.

How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

As an alternative, you can specify a value for *keystore.recipient.alias* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for *keystore.enc.csf.key* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy.

How to Set Up the Web Service Client at Design Time

This policy uses symmetric key technology, which is an encryption method that uses the same shared key to encrypt and decrypt data. The symmetric key is used to sign the message.

Configure the policy assertion for message signing, message encryption, or both.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

Example 10–3 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.

oracle/wss11_username_token_with_message_protection_service_policy

This policy enforces message-level protection (that is, message integrity and message confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following policy assertion: oracle/wss11_username_token_with_message_protection_service_template. See "oracle/wss11_username_token_with_message_protection_service_template" on page C-46 for more information about the assertion.

Settings You Can Change

See Table C–48.

Properties You Can Configure

See Table C–50. You also have the option to override the *keystore.enc.csf.key* server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Store the trusted certificate that corresponds to the client’s private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.enc.csf.key* server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up WebLogic Server

Use the WebLogic Server Administration Console to add an Authentication provider to the active security realm for the WebLogic domain in which the Web service is deployed, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14.

oracle/wss11_x509_token_with_message_protection_client_policy

This policy provides message-level protection and certificate-based authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following policy assertion: oracle/wss11_x509_token_with_message_protection_client_template. See "oracle/wss11_x509_token_with_message_protection_client_template" on page C-46 for more information about the assertion.

Settings You Can Change

See Table C–51.

Properties You Can Configure

See Table C–52.

How to Set Up the Web Service Client

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the Web service keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

The Web service's base64-encoded public certificate is published in the WSDL for use by the Web service client, as described in "Using Service Identity Certification Extension" on page 9-10.

As an alternative, you can specify a value for *keystore.recipient.alias* on the **Configurations** page, or override it on a per-client basis using the **Security Configuration Details** control when you attach the policy. The keystore recipient alias specifies the alias used to look up the public key in the keystore when retrieving a key for encryption of outbound SOAP messages.

You can specify a value for *keystore.sig.csf.key* and *keystore.enc.csf.key* on the **Configurations** page, or override them on a per-client basis using the **Security Configuration Details** control when you attach the policy.

How to Set Up the Web Service Client at Design Time

This policy requires you to set up the Web service client keystore, as described in "Setting Up the Web Service Client Keystore at Design Time" on page 9-8. The policy specifically requires that the client's and Web service's respective keystores already contain digital certificates containing each other's public key.

The Web service client needs to provide valid X.509 authentication credentials in the SOAP message through the WS-Security binary security token.

See "Using Client Programmatic Configuration Overrides" on page 10-58 for a description of the configuration settings you can override.

Configure the policy assertion for message signing, message encryption, or both.

Example 10–3 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.

oracle/wss11_x509_token_with_message_protection_service_policy

This policy enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy contains the following policy assertion: oracle/wss11_x509_token_with_message_protection_service_template. See "oracle/wss11_x509_token_with_message_protection_service_template" on page C-48 for more information about the assertion.

Settings You Can Change

See Table C–51.

Properties You Can Configure

See Table C–53. You also have the option to override the *keystore.enc.csf.key* server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up Oracle Platform Security Services (OPSS)

Configure the policy assertion for message signing, message encryption, or both.

This policy requires you to set up the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

Store the trusted certificate that corresponds to the client's private key (used to sign the message) in the keystore. You also need to store the service's private key in the keystore for decrypting the message, and the CA root certificate.

You must store the password for the decryption key in the credential store, as described in "Configuring the Credential Store Provider" on page 9-12. Use *keystore.enc.csf.key* as the key name.

You also have the option to override the *keystore.enc.csf.key* server-side configuration property, as described in "Attaching Web Service Policies Permitting Overrides" on page 8-14.

How to Set Up WebLogic Server

You need to configure the Authentication provider, as described in "Configuring an Authentication Provider in WebLogic Server" on page 9-14, and make sure that you provide the X.509 callback information for this provider.

Authorization Policies and Configuration Steps

Frequently, authentication is the first step of determining whether a user should be given access to a Web service. After the user is authenticated, the second step is to verify that the user is authorized to access the Web service. This is accomplished using an authorization policy. You can create an authorization policy using the *binding_authorization_template* or the *component_authorization_template* assertion templates.

Policies created with these templates perform role- or permission-based access control (RBAC) and check that the authenticated user has been granted one of the roles or permissions allowed access to the Web service.

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

Note: The authorization polices can follow any authentication policy where the subject is established.

You cannot attach both a permitall and denyall policy to the same Web service.

Determining Which Resources to Protect

The authorization policies provide the following properties that you can use to specify which resources you want the policy to protect. Not all of the predefined policies feature all of the properties.

- Constraint Pattern -- Reserved for future use.
- Action Pattern -- The Web service operation for which permission-based checks are performed. This value can be a comma-separated list of values. This field accepts wildcards. * means all Web service operations.

  The valid values for Action Pattern are determined by the Web service methods. For example, if the Web service method is *validate(amountAvailable)*, enter the Action Pattern as *validate*.
- Resource Pattern -- The name of the resource for which permission-based checks are performed. This field accepts wildcards, and the default is * for all resources in the Web services protected by the policy.

  By convention you enter the Resource Pattern as (namespace of Web service + Web service name).

  For example, if the namespace of the Web service is *http://project11* and the Web service name is *CreditValidation*, you would enter the Resource Name as *http://project11/CreditValidation*.

  If you specify a specific Resource Pattern, the policy is enforced only for those Web services that match the criteria. That is, entering a specific Resource Pattern limits the scope of the authorization policy. This condition also applies if you have bulk-attached this authorization policy to multiple subjects. The default of * protects all resources (namespace of Web service + Web service name) of the bulk-attached Web services.
- Permission Check Class -- By default, it is *oracle.wsm.security.WSFunctionPermission*. The class must be in the classpath.
- Authorization Setting -- Possible values are Permit All, Deny All, and Selected Roles. If you choose Selected Roles, you must then select from the enterprise (Global) roles defined in WebLogic Server, which may include the following:
  - AdminChannelUser
  - Anonymous
  - AppTester
  - CrossDomainConnector
  - Deployer
  - Monitor
  - Operator
  - OracleSystemRole

How Authorization Permissions Are Determined

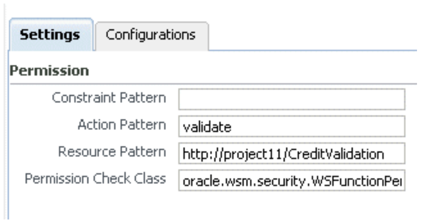
Conceptually, determining whether an authenticated subject is authorized to access a particular resource protected by a Web service policy has two parts that work in tandem.

- The **Resource Pattern** and **Action Pattern** parameters on the Policy Settings page for the policy determine what resources are being protected by the policy, as shown in Figure 10–2. (You can also override these properties, as described in "Configuring Server-Side Override Properties for Authorization Policies" on page 8-16.)

You have the option to change the *Permission Check Class* configuration property for the policy, which identifies the permission class as per JAAS standards. The permission class must be available in the application or server classpath.

The custom permission class must extend the abstract *Permission* class and implement the *Serializable* interface. See the Javadoc at `http://java.sun.com/j2se/1.5.0/docs/api/java/security/Permission.html`. The default is *oracle.wsm.security.WSFunctionPermission*.

Figure 10–2 The Permission Settings for a Policy

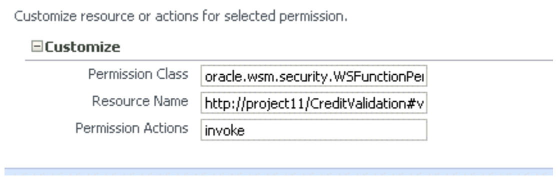


Permission

Constraint Pattern	
Action Pattern	validate
Resource Pattern	http://project11/CreditValidation
Permission Check Class	oracle.wsm.security.WSFunctionPe

- The OPSS Application Policies page specifies whether the authenticated subject has invoke access to the **Resource Name** listed there, as shown in Figure 10–3.

Figure 10–3 Adding a Permission on the OPSS Create Application Grant Page



Customize resource or actions for selected permission.

Customize

Permission Class	oracle.wsm.security.WSFunctionPe
Resource Name	http://project11/CreditValidation#v
Permission Actions	invoke

OPSS uses the Policy Settings page for the Web service to determine which resources require an authorization check. Then, access to the resource is allowed if the authenticated subject has been granted *WSFunctionPermission* (or other permission) for that resource via OPSS.

Note: If you changed the *Permission Check Class* configuration property for the policy to a custom class, use the custom class here as well.

Consider further the example shown in Figure 10–2 and Figure 10–3.

On the Policy Settings page, assume that you specify the following to protect the *validate* method of the *http://project11/CreditValidation* Web service:

```
Action pattern:         validate
Resource pattern:        http://project11/CreditValidation
Permission Check Class   oracle.wsm.security.WSFunctionPermission
```

Then, on the OPSS Application Policies page, you would use *http://project11/CreditValidation#validate* for the **Resource Name** to specify that the authenticated subject has permission to invoke this resource:

```
Permission Class: oracle.wsm.security.WSFunctionPermission
Resource Name:    http://project11/CreditValidation#validate
Permissions Action:  invoke
```

You can grant the *WSFunctionPermission* permission to a user, a group, or an application role. If you grant *WSFunctionPermission* to a user or group it will apply to all applications that are deployed in the domain.

OPSS Resource Name Can Include Operation Name

In previous releases of Fusion Middleware Control, the **Resource Name** on the OPSS Application Policies page was determined by *name-space-of-webservice/ServiceName*. For example, if the name space of a Web service was *http://project1/* and the service name was *CreditValidation*, the **Resource Name** would have been *http://project1/CreditValidation*. You could also use an asterisk (*) wildcard for providing permission to all the actions or all resources.

In this release, the resource target of the *WSFunctionPermission* is enhanced to include the actual Web service operation name. The syntax for the **Resource Name** is now *name-space-of-webservice/servicename#[operation name]*. (For a component it is *compositename/componentname#[operation name]*.)]

You must now include at least the *name-space-of-webservice/service name*. That is, you can no longer use an asterisk (*) wildcard for providing permission to all the actions or all resources.

Instead, to specify all operations for a Web service, simply leave the operation name blank. For example, *name-space-of-webservice/servicename#*

Permission Action is always *invoke*.

oracle/binding_authorization_denyall_policy

This policy provides a simple role-based authorization policy based on the authenticated subject.

This policy denies all users with any role.

This policy should follow an authentication policy where the subject is established and can be attached to any SOAP-based endpoint.

You must have already configured a WebLogic Authentication provider, as described in "Configure Authentication Providers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

This policy contains the following policy assertion: oracle/binding_authorization_template

See "oracle/binding_authorization_template" on page C-49 for more information about the assertion.

Settings You Can Change

See Table C–55.

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**.

Properties You Can Configure

None defined.

How to Set Up Oracle Platform Security Services (OPSS)

If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. You use the WebLogic Server Administration Console to grant a role to a user or group, as described in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication Providers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

oracle/binding_authorization_permitall_policy

This policy provides a simple role-based authorization policy based on the authenticated subject.

This policy permits all users with any roles.

This policy should follow an authentication policy where the subject is established and can be attached to any SOAP-based endpoint.

You must have already configured a WebLogic Authentication provider, as described in "Configure Authentication Providers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

This policy contains the following policy assertion: oracle/binding_authorization_template. See "oracle/binding_authorization_template" on page C-49 for more information about the assertion.

Settings You Can Change

See Table C–55.

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**.

Properties You Can Configure

None defined.

How to Set Up Oracle Platform Security Services (OPSS)

If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. You use the WebLogic Server Administration Console to grant a role to a user or group, as described in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication providers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

oracle/binding_permission_authorization_policy

This policy provides a permission-based authorization policy based on the authenticated subject.

This policy ensures that the subject has permission to perform the operation. To do this, the Authorization Policy executor leverages OPSS to check if the authenticated subject has been granted *oracle.wsm.security.WSFunctionPermission* (or whatever permission class is specified in *Permission Check Class*) using the *Resource Pattern* and *Action Pattern* as parameters.

This policy should follow an authentication policy where the subject is established and can be attached to any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/binding_permission_authorization_template. See "oracle/binding_permission_authorization_template" on page C-50 for more information about the assertion.

Settings You Can Change

See Table C–56.

Attributes You Can Configure

You have the option to change the *permission_class* configuration property for the policy, which identifies the permission class as per JAAS standards. The permission class must be available in the application or server classpath.

The custom permission class must extend the abstract *Permission* class and implement the *Serializable* interface. See the Javadoc at `http://java.sun.com/j2se/1.5.0/docs/api/java/security/Permission.html`.

The default is *oracle.wsm.security.WSFunctionPermission*.

How to Set Up Oracle Platform Security Services (OPSS)

Use Fusion Middleware Control to grant the *WSFunctionPermission* (or other) permission to the user, group, or application that will attempt to authenticate to the Web service.

You have the option to change the *permission_class* configuration property for the policy, which identifies the permission class as per JAAS standards. The class must be available in the server classpath. The default is *oracle.wsm.security.WSFunctionPermission*.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication Providers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

oracle/component_authorization_denyall_policy

This policy provides a simple role-based authorization policy based on the authenticated subject.

This policy denies all users with any roles.

You must have already configured a WebLogic Authentication provider, as described in "Configure Authentication providers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

This policy should follow an authentication policy where the subject is established and can be attached to any SCA-based endpoint.

This policy contains the following policy assertion: oracle/component_authorization_template. See "oracle/component_authorization_template" on page C-51 for more information about the assertion.

Settings You Can Change

See Table C–57.

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**.

Properties You Can Configure

None defined.

How to Set Up Oracle Platform Security Services (OPSS)

If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. You use the WebLogic Server Administration Console to grant a role to a user or group, as described in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication providers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

oracle/component_authorization_permitall_policy

This policy provides a simple role-based authorization policy based on the authenticated subject.

This policy permits all users with any roles.

You must have already configured a WebLogic Authentication provider, as described in "Configure Authentication providers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

It should follow an authentication policy where the subject is established and can be attached to any SCA-based endpoint.

This policy contains the following policy assertion: oracle/component_authorization_template. See "oracle/component_authorization_template" on page C-51 for more information about the assertion.

Settings You Can Change

See Table C–57.

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**.

Properties You Can Configure

None defined.

How to Set Up Oracle Platform Security Services (OPSS)

If you specify one or more of the WebLogic Server enterprise roles, the authenticated subject must already have that role. You use the WebLogic Server Administration

Console to grant a role to a user or group, as described in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication providers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

oracle/component_permission_authorization_policy

This policy provides a permission-based authorization policy based on the authenticated subject.

This policy ensures that the subject has permission to perform the operation. To do this, the Authorization Policy executor leverages OPSS to check if the authenticated subject has been granted *oracle.wsm.security.WSFunctionPermission* (or whatever permission class is specified in *Permission Check Class*) using the *Resource Pattern* and *Action Pattern* as parameters. *Resource Pattern* and *Action Pattern* are used to identify if the authorization assertion is to be enforced for this particular request. Access is allowed if the authenticated subject has been granted *WSFunctionPermission*.

You can grant the *WSFunctionPermission* permission to a user, a group, or an application role. If you grant *WSFunctionPermission* to a user or group it will apply to all applications that are deployed in the domain.

This policy should follow an authentication policy where the subject is established and can be attached to any SCA-based endpoint.

This policy contains the following policy assertion: oracle/component_permission_authorization_template. See "oracle/component_permission_authorization_template" on page C-52 for more information about the assertion.

Settings You Can Change

See Table C–58.

Properties You Can Configure

None defined.

How to Set Up Oracle Platform Security Services (OPSS)

Use Fusion Middleware Control to grant the *WSFunctionPermission* permission to the user, group, or application that will attempt to authenticate to the Web service.

You must configure a WebLogic Authentication provider, as described in "Configure Authentication providers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

WS-Addressing Policies and Configuration Steps

The Web Services Addressing (WS-Addressing) specification (`http://www.w3.org/TR/ws-addr-core/`) provides transport-neutral mechanisms to address Web services and messages. In particular, the specification defines a number of XML elements used to identify Web service endpoints and to secure end-to-end endpoint identification in messages.

This section describes the predefined WS-Addressing policies.

oracle/wsaddr_policy

This policy causes the platform to check inbound messages for the presence of WS-Addressing headers conforming to the W3C 2005 Final WS-Addressing Policy standard. In addition, it causes the platform to include a WS-Addressing header in outbound SOAP messages.

How to Set Up the Web Service Client

No configuration is needed.

How to Set Up the Web Service Client at Design Time

Configure WS-Addressing for the Web service client as described in the *Web Services Addressing 1.0 - SOAP Binding* specification (`http://www.w3.org/TR/ws-addr-soap/`).

How to Set Up Oracle Platform Security Services (OPSS)

No configuration is needed.

MTOM Attachment Policies and Configuration Steps

This section describes the predefined MTOM policies.

oracle/wsmtom_policy

SOAP Message Transmission Optimization Mechanism/XML-binary Optimized Packaging (MTOM/XOP) defines a method for optimizing the transmission of XML data of type xs:base64Binary or xs:hexBinary in SOAP messages.

The Message Transmission Optimization Mechanism (MTOM) policy rejects inbound messages that are not in MTOM format and verifies that outbound messages are in MTOM format.

MTOM refers to specifications `http://www.w3.org/TR/2005/REC-soap12-mtom-20050125` and `http://www.w3.org/Submission/2006/SUBM-soap11mtom10-20060405` for SOAP 1.2 and SOAP 1.1 bindings, respectively.

How to Set Up the Web Service Client

No configuration is required.

How to Set Up the Web Service Client at Design Time

To enable MTOM on the client of the Web service, pass the javax.xml.ws.soap.MTOMFeature as a parameter when creating the Web service proxy or dispatch, as illustrated in the following example.

```
package examples.webservices.mtom.client;
import javax.xml.ws.soap.MTOMFeature;
public class Main {
  public static void main(String[] args) {
    String FOO = "FOO";
    MtomService service = new MtomService()
    MtomPortType port = service.getMtomPortTypePort(new MTOMFeature());
    String result = null;
    result = port.echoBinaryAsString(FOO.getBytes());
    System.out.println( "Got result: " + result );
    }
}
```

How to Set Up Oracle Platform Security Services (OPSS)

No configuration is required.

Reliable Messaging Policies and Configuration Steps

WS-ReliableMessaging makes message exchanges reliable. It ensures that messages are delivered reliably between distributed applications regardless of software component, system, or network failures. Ordered delivery is assured and automatic retransmission of failed messages does not have to be coded by each client application.

Consider using reliable messaging if your Web service is experiencing the following problems:

- network failures or dropped connections
- messages are lost in transit
- messages are arriving at their destination out of order

WS-ReliableMessaging considers the source and destination of a message to be independent of the client/server model. That is, the client and the server can each act simultaneously as both a message source and destination on the communications path.

This section describes the predefined Reliable Messaging policies.

WS-RM Policy Properties

Table 10–1 lists the properties that you can set for the WS-RM policies.

Table 10–1 WS-RM Policy Properties

Property Name	Description	Default Value Used by Policy	Possible Values
DeliveryAssurance	Delivery assurance. The following defines the delivery assurance types: ■ At Most Once—Messages are delivered at most once, without duplication. ■ At Least Once—Every message is delivered at least once. It is possible that some messages are delivered more than once. ■ Exactly Once—Every message is delivered exactly once, without duplication. ■ Messages are delivered in the order that they were sent. This delivery assurance can be combined with one of the preceding three assurances.	InOrder	InOrder AtLeastOnce AtLeastOnceInOrder ExactlyOnce ExactlyOnceInOrder AtMostOnce AtMostOnceInOrder
StoreType	Type of message store.	InMemory	InMemory FileSystem (not fully supported) JDBC
StoreName	Name of the message store.	oracle	String value

Table 10–1 (Cont.) WS-RM Policy Properties

Property Name	Description	Default Value Used by Policy	Possible Values
jdbc-connection-name	JNDI reference to a JDBC data source. This field is valid only if StoreType is set to JDBC. This value takes precedence over jdbc-connection-url. The username and password will be used if both are present.	jdbc/MessagesStore	Valid JDBC store
InactivityTimeout	Amount of time, in milliseconds, that can elapse between message exchanges associated with a particular WS-ReliableMessaging sequence. Once this value is reached, the sequence will be terminated and discarded automatically.	600000	The amount of time in milliseconds.
BaseRetransmissionInterval	Interval, in milliseconds, that the source endpoint waits after transmitting a message and before it retransmits the message if it receives no acknowledgment for that message.	3000	The amount of time in milliseconds.

oracle/wsrm10_policy

This policy provides support for version 1.0 of the Web Services Reliable Messaging protocol. This policy can be attached to any SOAP-based client or endpoint.

How to Set Up the Web Service Client

The Web service client will automatically detect the WSDL policy assertions at runtime and use them to enable the advertised version of WS-RM on the client.

How to Set Up the Web Service Client at Design Time

For multi-message sequences, the client code must include explicit invocations of methods for delimiting sequence boundaries. Otherwise, every message is wrapped in its own sequence

Edit the client to enable a reliable messaging session for the messages sent to the service. The *oracle.webservices.rm.client.RMSessionLifecycle* interface provides the client with a mechanism for demarcating WS-RM sequence boundaries.

Example 10–4 illustrates sample WS-RM client code. In the code, a new TestService is created. The TestPort, through which the client will communicate with the service, is retrieved. The port object is cast to a *RMSessionLifecycle* object and a reliable messaging session is opened on it (*openSession*). After the messages are sent to the service, the session is closed (*closeSession*).

Example 10–4 Sample WS-Rm Client Code

```
public class ClientServlet extends HttpServlet {

    public void doGet(HttpServletRequest request,
                HttpServletResponse response) throws ServletException,
                                                     IOException {

        int num1 =  Integer.parseInt(request.getParameter("num1"));
        int num2 =  Integer.parseInt(request.getParameter("num2"));
        String outputStr = null;
        TestService service = new TestService();
        Test port = service.getTestPort();

        try {
        ((RMSessionLifecycle) port).openSession();
            outputStr = port.hello(inputStr);
        } catch (Exception e) {
            e.printStackTrace();
            outputStr = e.getMessage();
        } finally {
        ((RMSessionLifecycle) port).closeSession();
            response.getOutputStream().write(outputStr.getBytes());
        }
    }
}
```

How to Set Up Oracle Platform Security Services (OPSS)

No additional configuration is required.

oracle/wsrm11_policy

This policy provides support for version 1.1 of the Web Services Reliable Messaging protocol. This policy can be attached to any SOAP-based client or endpoint.

How to Set Up the Web Service Client

The Web service client will automatically detect the WSDL policy assertions at runtime and use them to enable the advertised version of WS-RM on the client.

How to Set Up the Web Service Client at Design Time

For multi-message sequences, the client code must include explicit invocations of methods for delimiting sequence boundaries. Otherwise, every message is wrapped in its own sequence

Edit the client to enable a reliable messaging session for the messages sent to the service. The *oracle.webservices.rm.client.RMSessionLifecycle* interface provides the client with a mechanism for demarcating WS-RM sequence boundaries.

Example 10–4 illustrates a servlet client. In the code, a new TestService is created. The TestPort, through which the client will communicate with the service, is retrieved. The port object is cast to a *RMSessionLifecycle* object and a reliable messaging session is opened on it (*openSession*). After the messages are sent to the service, the session is closed (*closeSession*).

How to Set Up Oracle Platform Security Services (OPSS)

No additional configuration is required.

Management Policies and Configuration Steps

This section describes the predefined Management policies.

oracle/log_policy

This policy causes the request, response, and fault messages to be sent to a message log.

This policy contains the following policy assertion: *oracle/log_template*. See "oracle/security_log_template" on page C-53 for more information about the assertion.

Settings You Can Change

See Table C–60.

Properties You Can Configure

None defined.

How to Set Up the Web Service or Client

Determine whether you want to log messages for the request and response, based on the following categories:

- all
- header
- SOAP body
- SOAP envelope

How to Set Up Oracle Platform Security Services (OPSS)

Messages are logged to the message log for the domain.

To view the message log

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you want to see the logged messages.   Select the domain.
2. Using Fusion Middleware Control, click **Weblogic Domain**, then **Logs** and then **View Log Messages**.

Attaching Policy Files to Web Services and Clients

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

Post-deployment, you attach security and management policies to SOA composites, ADF, and WebCenter applications using the Oracle Enterprise Manager Fusion Middleware Control. This method provides the most power and flexibility because it moves Web service security to the control of the security administrator.

At design time, Oracle JDeveloper automates ADF and SOA client policy attachment. Or, you can attach Oracle WSM security and management policies to applications programmatically. You typically do this using your favorite IDE, such as Oracle JDeveloper.

Either way, the client-side policy must be the equivalent of the one associated with the Web service. If the two files are different, and there is a conflict in the assertions contained in the files, then the invoke of the Web service operation returns an error.

For example, if the oracle/wss_http_token_over_ssl_service_policy policy requires mutual authentication, the client policy must also be set for mutual authentication.

For the predefined policies, both client and Web service policies are included. If you create a new policy, generating the policy as described in "Creating Web Service Policies" on page 7-4 increases the likelihood that the client policy will work with the service policy.

Using Client Programmatic Configuration Overrides

"Attaching Client Policies Permitting Overrides" on page 8-13 describes the policy configuration override feature that allows you to specify certain Web service client configuration information when you attach a policy. However, you can also override this configuration information programmatically at design time. This section describes client programmatic overrides.

Table 10–2 shows the properties you can set via programmatic configuration overrides for a given policy. Example 10–5 shows an example of setting these properties from a program.

Table 10–2 Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_CSF_KEY	Gets the username and password corresponding to the csf-key specified in the credential store if the credential store is available to the client.	oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss_username_token_client_policy oracle/wss_username_token_over_ssl_client_policy oracle/wss_username_token_with_digestpassword_client_policy oracle/wss10_username_id_propagation_with_msg_protection_client_policy oracle/wss_http_token_client_policy oracle/wss_http_token_over_ssl_client_policy

Table 10–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_LOCATION	This property sets the location of the keystore file. If provided, this value will override any statically configured value. Type: java.lang.String	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy oracle/wss11_kerberos_token_with_message_protection_client_policy oracle/wss11_message_protection_client_policy oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss11_x509_token_with_message_protection_client_policy

Table 10–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_TYPE	This property sets the type of keystore file. If provided, this value will override any statically configured value. Type: java.lang.String Default is JKS.	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy oracle/wss11_kerberos_token_with_message_protection_client_policy oracle/wss11_message_protection_client_policy oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss11_x509_token_with_message_protection_client_policy

Table 10–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KEYSTORE_PASSWORD	This property sets the password of the keystore file. If provided, this value will override any statically configured value. Type: java.lang.String	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy oracle/wss11_kerberos_token_with_message_protection_client_policy oracle/wss11_message_protection_client_policy oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss11_x509_token_with_message_protection_client_policy

Table 10–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	**Description**	**Applies to These Policies**
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_SIG_KEY_ALIAS	This property sets the alias of the key within the keystore that will be used for digital signatures. If provided, this value will override any statically configured value. Type: java.lang.String For WSS11 policies, this property is used only in the case of mutual authentication.	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy oracle/wss11_kerberos_token_with_message_protection_client_policy oracle/wss11_message_protection_client_policy oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss11_x509_token_with_message_protection_client_policy

Table 10–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	**Description**	**Applies to These Policies**
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_SIG_KEY_PASSWORD	This property sets the password for the alias of the key within the keystore that will be used for digital signatures. If provided, this value will override any statically configured value. Type: java.lang.String For WSS11 policies, this property is used only in the case of mutual authentication.	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy oracle/wss11_kerberos_token_with_message_protection_client_policy oracle/wss11_message_protection_client_policy oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss11_x509_token_with_message_protection_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_ENC_KEY_ALIAS	This property sets the alias of the key within the keystore that will be used to decrypt the response from the service. If provided, this value will override any statically configured value. Type: java.lang.String Not used in WSS11 policies.	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy

Table 10–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_ENC_KEY_PASSWORD	This property sets the password for the key within the keystore that will be used for decryption. If provided, this value will override any statically configured value. Type: java.lang.String Not used in WSS11 policies.	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_RECIPIENT_KEY_ALIAS	This property sets the alias for the recipient's public key that is used to encrypt type outbound message. If provided this value will override any static configuration value. Type: java.lang.String	oracle/wss10_message_protection_client_policy oracle/wss10_saml_hok_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_integrity_client_policy oracle/wss10_saml_token_with_message_protection_client_policy oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy oracle/wss10_username_token_with_message_protection_client_policy oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy oracle/wss10_x509_token_with_message_protection_client_policy oracle/wss11_kerberos_token_with_message_protection_client_policy oracle/wss11_message_protection_client_policy oracle/wss11_saml_token_with_message_protection_client_policy oracle/wss11_username_token_with_message_protection_client_policy oracle/wss11_x509_token_with_message_protection_client_policy

Table 10–2 (Cont.) Properties Set Via Programmatic Configuration Overrides

Property List	Description	Applies to These Policies
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_SUBJECT_PRECEDENCE	In case of SAML client policies, set this property to false if there is a need to use a client-specified username rather than subject.	Applies to all of the SAML client policies listed in "Configuring SAML" on page 9-16.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_SAML_ISSUER_NAME	This property sets the SAML issuer name when trying access a service that is protected using SAML mechanism. If provided this value will override any static configuration value. Type: java.lang.String	Applies to all of the SAML client policies listed in "Configuring SAML" on page 9-16.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_INCLUDE_USER_ROLES	This property sets the user roles in a SAML assertion.	Applies to all of the SAML client policies listed in "Configuring SAML" on page 9-16.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSM_SAML_ASSERTION_FILE_NAME	For SAML HOK policies, this file contains the assertion	Applies to all of the SAML client policies listed in "Configuring SAML" on page 9-16.
oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_KERBEROS_SERVICE_PRINCIPAL	This property sets the service principal name when trying access a service that is protected using the Kerberos mechanism. If provided this value will override any static configuration value. Type: java.lang.String	oracle/wss11_kerberos_token_with_message_protection_client_policy

Configuration Override Example

Example 10–5 shows an example of a Web service client overriding the keystore and username/password.

If you need to clear an overridden configuration property, set it to an empty string.

Before you clear it, remember that other policies could be using the same property. The properties are client-specific and there could be multiple policies that are attached to the same client that use the same property.

Example 10–5 Overriding the Keystore and Username/Password

```
package example;
import oracle.wsm.security.utils.SecurityConstants;
public class MyClientJaxWs {
    public static void main(String[] args) {
        try {
            URL serviceWsdl = new URL("http://localhost/myApp/myPort?WSDL");
            QName serviceName = new QName("MyNamespace", "MyService");
            Service service = Service.create(serviceWsdl, serviceName);
            MyInterface proxy = service.getPort(MyInterface.class);
            RequestContext context = ((BindingProvider)proxy).getRequestContext();
            context.put(oracle.webservices.ClientConstants.CLIENT_CONFIG, new
File( "c:/dat/client-pdd.xml" ) );
            context.put(BindingProvider.USERNAME_PROPERTY, getCurrentUsername() );
            context.put(BindingProvider.PASSWORD_PROPERTY, getCurrentPassword() );
            context.put(SecurityConstants.ClientConstants.WSS_KEYSTORE_LOCATION,
"c:/mykeystore.jks");
            context.put(SecurityConstants.ClientConstants.WSS_KEYSTORE_PASSWORD,
"keystorepassword" );
            context.put(SecurityConstants.ClientConstants.WSS_KEYSTORE_TYPE, "JKS"
);
            context.put(SecurityConstants.ClientConstants.WSS_SIG_KEY_ALIAS, "your
signature alias" );
            context.put(SecurityConstants.ClientConstants.WSS_SIG_KEY_PASSWORD,
"your signature password" );
            context.put(SecurityConstants.ClientConstants.WSS_ENC_KEY_ALIAS, "your
encryption alias" );
            context.put(SecurityConstants.ClientConstants.WSS_ENC_KEY_PASSWORD,
"your encryption password" );
            System.out.println(proxy.myOperation("MyInput"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

In Example 10–5, the contents of *c:/dat/client-pdd.xml* referenced might be as follows:

```
! -- The contents of c:/dat/client-pdd.xml file mentioned above -- >
<oracle-webservice-clients>
  <webservice-client>
    <port-info>
      <policy-references>
        <policy-reference uri="management/Log_Msg_Policy" category="management"/>
        <policy-reference uri="oracle/wss10_username_token_with_message_
protection_client_policy" category="security"/>
      </policy-references>
    </port-info>
  </webservice-client>
</oracle-webservice-clients>
```

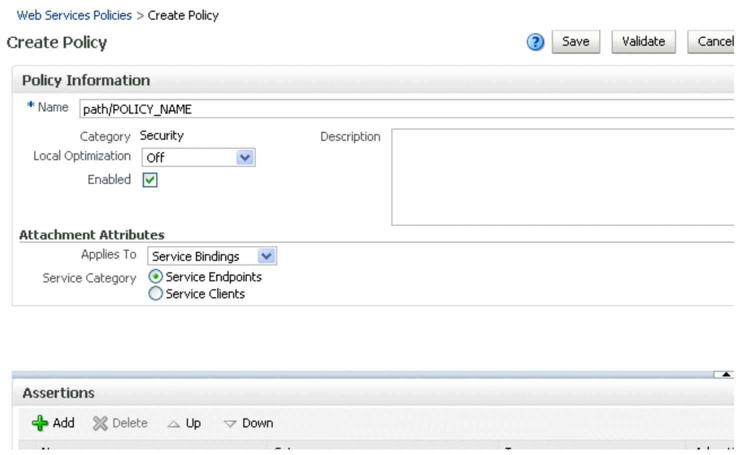
Configuring Local Optimization

Oracle WSM supports a SOA local optimization feature for composite-to-composite invocations in which the reference of one composite specifies a Web service binding to a second composite. Both composites must be running in the same container.

The optimization control is available when you create or edit a policy, as shown in Figure 10–4, and it provides for the following:

- HTTP is not called
- SOAP/Normalized Message conversion is not needed

Figure 10–4 Local Optimization Control When Creating a Policy


Web Services Policies > Create Policy

Create Policy

Policy Information

* Name: path/POLICY_NAME

Category: Security

Local Optimization: Off

Enabled: ✓

Description:

Attachment Attributes

Applies To: Service Bindings

Service Category: ● Service Endpoints ○ Service Clients

Assertions

If there is a policy attached to the Web service, the policy may not be invoked if this optimization is used. Therefore, for each policy you need to decide whether you want to use the local optimization.

There are three possible settings for the Local Optimization control: On, Off, and Check Identity:

- On -- Optimization is turned on.
- Off -- 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.

Table 10–3 shows the predefined policies, and describes how each policy implements the local optimization feature.

Table 10–3 Default Optimization Setting of Predefined Policies

Policy Name	Default Optimization Setting
oracle/wsaddr_policy	On
oracle/binding_authorization_denyall_policy	Always Off
oracle/binding_authorization_permitall_policy	Always Off
oracle/binding_permission_authorization_policy	Off
oracle/component_authorization_denyall_policy	Always Off. (Does not apply to bindings.)
oracle/component_authorization_permitall_policy	Always Off. (Does not apply to bindings.)
oracle/component_permission_authorization_policy	Off

Table 10–3 (Cont.) Default Optimization Setting of Predefined Policies

Policy Name	Default Optimization Setting
oracle/log_policy	On
oracle/wsmtom_policy	On
oracle/wss_http_token_client_policy	Check Identity
oracle/wss_http_token_service_policy	Check Identity
oracle/wss_http_token_over_ssl_client_policy	Check Identity
oracle/wss_http_token_over_ssl_service_policy	Check Identity
oracle/wss11_kerberos_token_client_policy	Check Identity
oracle/wss11_kerberos_token_service_policy	Check Identity
oracle/wss_username_token_client_policy	Check Identity
oracle/wss_username_token_service_policy	Check Identity
oracle/wss_username_token_over_ssl_client_policy	Check Identity
oracle/wss_username_token_over_ssl_service_policy	Check Identity
oracle/wss10_message_protection_client_policy	On
oracle/wss10_message_protection_service_policy	On
oracle/wss10_username_token_with_message_protection_client_policy	Check Identity
oracle/wss10_username_token_with_message_protection_service_policy	Check Identity
oracle/wss10_x509_token_with_message_protection_client_policy	Check Identity
oracle/wss10_x509_token_with_message_protection_service_policy	Check Identity
oracle/wss10_saml_token_with_message_protection_client_policy	Check Identity
oracle/wss10_saml_token_with_message_protection_service_policy	Check Identity
oracle/wss10_saml_token_client_policy	Check Identity

Table 10–3 (Cont.) Default Optimization Setting of Predefined Policies

Policy Name	Default Optimization Setting
oracle/wss10_saml_token_service_policy	Check Identity
oracle/wss10_username_id_propagation_with_msg_protection_client_policy	Check Identity
oracle/wss10_username_id_propagation_with_msg_protection_service_policy	Check Identity
oracle/wss11_message_protection_client_policy	On
oracle/wss11_message_protection_service_policy	On
oracle/wss11_username_token_with_message_protection_client_policy	Check Identity
oracle/wss11_username_token_with_message_protection_service_policy	Check Identity
oracle/wss11_x509_token_with_message_protection_client_policy	Check Identity
oracle/wss11_x509_token_with_message_protection_service_policy	Check Identity
oracle/wsrm10_policy	On
oracle/wsrm11_policy	On

11

Testing Web Services

This chapter includes the following sections:

- Testing Your Web Services
- Editing the Input Arguments as XML Source
- Enabling Authentication
- Enabling Quality of Service Testing
- Enabling HTTP Transport Options
- Stress Testing the Web Service Operation
- Disabling the Test Page for a Web Service

Testing Your Web Services

This section describes how to use the Fusion Middleware Control Test Web Service page to verify that you are receiving the expected results from the Web service.

The Test Web Service page allows you to test any of the operations exposed by a Web service. You can test Web services that are deployed on any accessible host; the Web service does not have to be deployed on this host.

Note: The Test Web Service page can parse WSDL URLs that contain ASCII characters only. If the URL contains non-ASCII characters, the parse operation fails. To test a Web service that has non-ASCII characters in the URL, allow your browser to convert the WSDL URL and use the resulting encoded WSDL URL in the Test Web Service page.

When testing Web services that use policies, the Oracle WSM component must be installed in the same domain that Fusion Middleware Control is being run. Otherwise, an invalid policy exception will be returned.

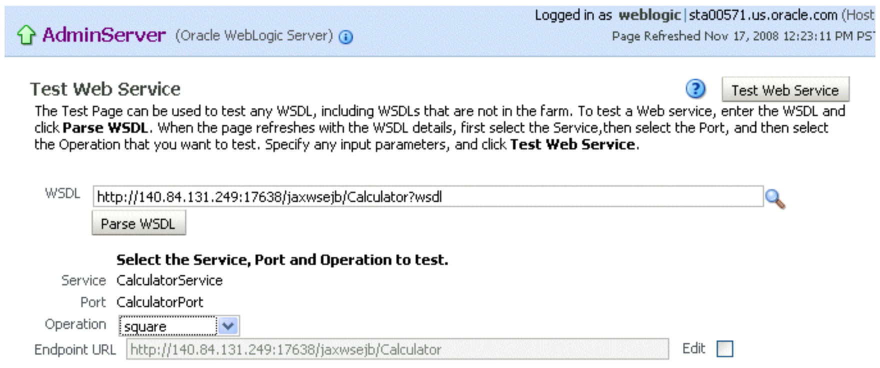
You can navigate to the Test Web Service page in many ways. This section describes one typical way to do so.

To test your Web service

1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to test a Web service.
2. Select the domain.

3. Using Fusion Middleware Control, click **WebLogic Domain** drop-down menu, **Web Services**, and then **Test Web Service**. The Test Web Service input page appears.
4. Enter the WSDL of the Web service you want to test and click **Parse WSDL**. If you do not know the WSDL, click the search link and select from the registered Web services, if any.
5. The **Test Web Service** page appears, as shown in Figure 11–1 and Figure 11–2.

Figure 11–1 Top Portion of Test Web Service Page



AdminServer (Oracle WebLogic Server)

Logged in as **weblogic** | sta00571.us.oracle.com (Host

Page Refreshed Nov 17, 2008 12:23:11 PM PS

Test Web Service

The Test Page can be used to test any WSDL, including WSDLs that are not in the farm. To test a Web service, enter the WSDL and click **Parse WSDL**. When the page refreshes with the WSDL details, first select the Service,then select the Port, and then select the Operation that you want to test. Specify any input parameters, and click **Test Web Service**.

WSDL: http://140.84.131.249:17638/jaxwsejb/Calculator?wsdl

Select the Service, Port and Operation to test.

Service: CalculatorService

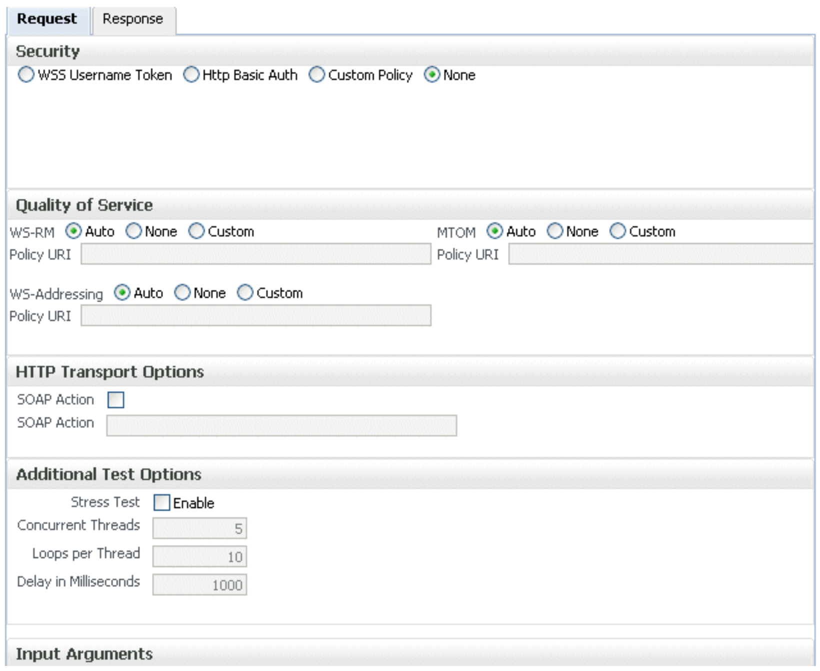
Port: CalculatorPort

Operation: square

Endpoint URL: http://140.84.131.249:17638/jaxwsejb/Calculator

Edit ☐

Figure 11–2 Bottom Portion of Test Web Service Page



Security

WSS Username Token Http Basic Auth Custom Policy [✓ None]

Quality of Service

WS-RM [✓ Auto] None Custom

Policy URI

MTOM [✓ Auto] None Custom

Policy URI

WS-Addressing [✓ Auto] None Custom

Policy URI

HTTP Transport Options

SOAP Action ☐

SOAP Action

Additional Test Options

Stress Test ☐ Enable

Concurrent Threads: 5

Loops per Thread: 10

Delay in Milliseconds: 1000

Input Arguments

6. Select the operation to perform during the test from the **Operation** control. The available operations are determined from the WSDL.

   To test a RESTful Web service, select the GET or POST service port operations.
7. If you want to change the Endpoint URL of the test, click **Edit** and make the change.
8. Select the **Request** tab if it is not already selected.

9. In the Security section, select the type of security token to verify. The security setting is not determined from a policy in the WSDL; you can specify the type of token you want to test. The default is None. If you do specify a username and password, they must exist and be valid for the WebLogic Server.

   When testing RESTful Web services, because the SOAP protocol is not used, the only security options are HTTP Basic Authentication or None.

10. In the Quality of Service section, specify whether you want to explicitly test a Reliable Messaging, WS-Addressing, or a MTOM policy.

    Note: This section is not available when testing RESTful Web services.

    In the default setting of Auto, WS-RM, WS-Addressing, and MTOM policies found in the WSDL are taken into consideration.

11. In the HTTP Transport section, the test mechanism uses the WSDL to determine whether a SOAP action is available to test.

    Note: This section is not available when testing RESTful Web services.

12. In the Additional Test Options section, set the **Stress Test** control if you want to invoke the Web service multiple times simultaneously. If you set this control, you can provide values for the stress test options or accept the defaults.

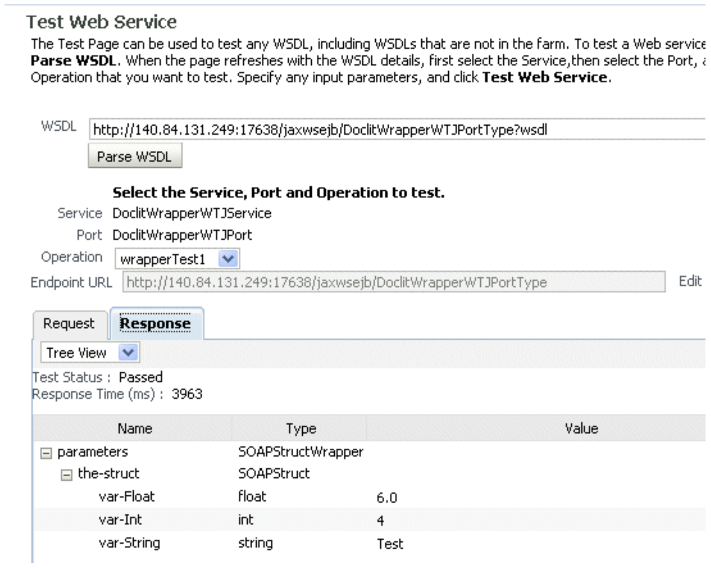
13. In the Input Arguments section, the parameters and type are determined from the WSDL, and require you to enter values of the correct type.

    You can view this section in Tree view or XML view.

14. Click **Test Web Service** to initiate the test.

15. If the test is successful, the **Test Status** field indicates *passed*, and the response time is displayed, as shown in Figure 11–3.

Figure 11–3 Successful Test



Test Web Service

The Test Page can be used to test any WSDL, including WSDLs that are not in the farm. To test a Web service
Parse WSDL. When the page refreshes with the WSDL details, first select the Service,then select the Port,
Operation that you want to test. Specify any input parameters, and click **Test Web Service**.

WSDL http://140.84.131.249:17638/jaxwsejb/DoclitWrapperWTJPortType?wsdl

Select the Service, Port and Operation to test.

Service DoclitWrapperWTJService
Port DoclitWrapperWTJPort
Operation wrapperTest1
Endpoint URL http://140.84.131.249:17638/jaxwsejb/DoclitWrapperWTJPortType

Test Status : Passed
Response Time (ms) : 3963

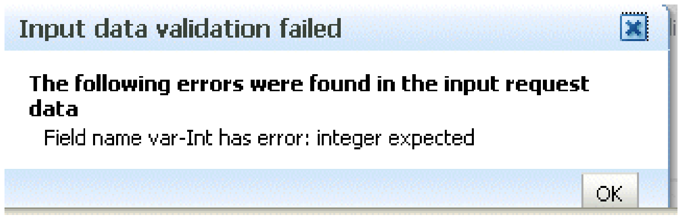
Name	Type	Value
parameters	SOAPStructWrapper	
the-struct	SOAPStruct	
var-Float	float	6.0
var-Int	int	4
var-String	string	Test

Note: The results on the **Response** tab are a simplified version of the standard Web service results.

---

16. If the test fails, an error message is displayed. For example, Figure 11–4 shows an error resulting from a type error in the *var-Int* parameter. In this particular instance, *string* data was entered when an *int* was expected.

Figure 11–4 Data Validation Error



Input data validation failed

The following errors were found in the input request data

Field name var-Int has error: integer expected

Editing the Input Arguments as XML Source

You can view the input arguments in a user-friendly form, or you can edit the XML source code directly. If you edit the XML source directly, you must enter valid XML. Use the drop-down list in the Input Arguments section of the page to toggle between **Tree View** and **XML View.**

Enabling Authentication

You can use the Test Page to test policies that use username tokens to authenticate users.

Note: Only policies that expect a username and password are supported by the test function, including custom policies. Policies that require certificates or other tokens are not supported.

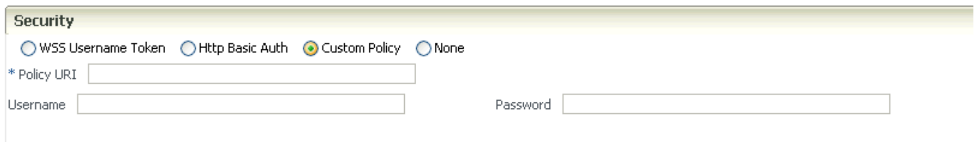
The security setting is not determined from a policy in the WSDL; you can specify the type of token you want to test. The default is None. If you do specify a username and password, they must exist and be valid.

The password must be passed in plain text. Authentication credentials may be supplied in the request by selecting one of the options in the Security section of the page (Figure 11–5). Select one of the following:

- **WSS-Username Token** – A WS-Security SOAP header is inserted. Username is required, and password is optional.
- **Http Basic Auth** – Username and password credentials are inserted in the HTTP transport header. Both the username and password are required.
- **Custom Policy** – A custom policy can be used to authenticate the user. You must specify the URI for the policy. The username and password are optional.
- **None** – No credentials are included.

Note: When testing RESTful Web services, because the SOAP protocol is not used, the only security options are HTTP Basic Authentication or None.

Figure 11–5 Security Parameters on the Web Services Test Page



Security

WSS Username Token [Http Basic Auth] [✓ Custom Policy] None

* Policy URI

Username

Password

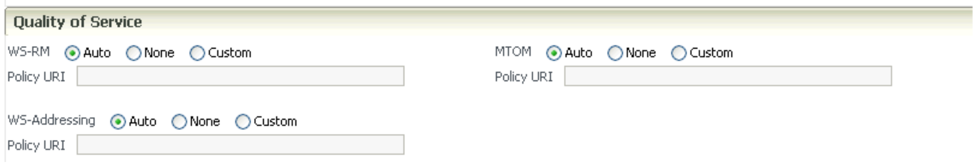
Enabling Quality of Service Testing

Note: This section is not applicable when testing RESTful Web services.

Three characteristics of Quality of Service (QoS) can be tested: reliable messaging (WS-RM), WS-Addressing, and Message Transmission Optimization Mechanism (MTOM) in the Quality of Service section of the Web Services Test Page (Figure 11–6). For each type of Quality of Service, there are three options:

- **Auto** – Execute the default behavior of the WSDL. For example, if **Auto** is selected for MTOM, and the WSDL contains a reference to an MTOM policy, the policy is enforced. If the WSDL does not contain a reference to an MTOM policy, then no MTOM policy is enforced.
- **None** – No policy for the specific QoS, even if it is included in the WSDL, is executed. For example, if **None** is selected for WS-RM, no reliable messaging policy is enforced. If the WSDL contains a reference to a reliable messaging policy, it is ignored.
- **Custom** – Enforce a custom policy. For example, if a WS-Addressing policy is referenced in the WSDL, this policy will be ignored, and the policy specified in **URI** will be used instead.
- **URI** – Specify the location of the policy to be enforced.

Figure 11–6 Quality of Service Parameters on Web Services Test Page



Quality of Service

WS-RM [✓ Auto] None Custom

Policy URI

WS-Addressing [✓ Auto] None Custom

Policy URI

MTOM [✓ Auto] None Custom

Policy URI

Enabling HTTP Transport Options

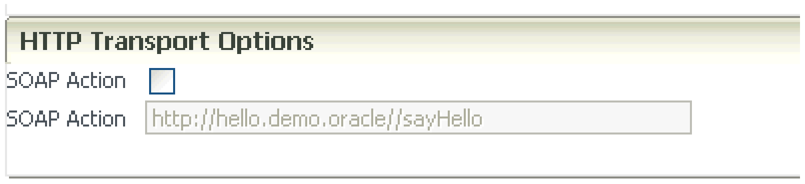
Note: This section is not applicable when testing RESTful Web services.

The test mechanism uses the WSDL to determine whether a SOAP action is available to test. If the WSDL soap:operation has a soapAction attribute, then this is displayed and **SOAP Action** is enabled.

When a request is sent with SOAP Action enabled, then the SOAP action HTTP header is sent.

To change this behavior, clear the SOAP Action box, in which case the HTTP header is not sent. Or, you can override the behavior by providing a different value in the SOAP Action text box. (You must already know the SOAP action that you want to test, and the syntax.)

Figure 11–7 HTTP Transport Options on Web Services Test Page



HTTP Transport Options

SOAP Action ☐

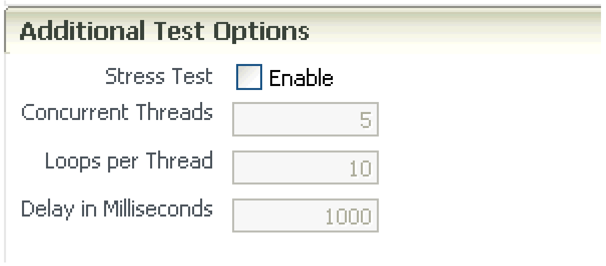
SOAP Action http://hello.demo.oracle//sayHello

Stress Testing the Web Service Operation

Select the Stress Test **Enable** check box (Figure 11–8) to display the options to create and configure a continuous series of invocations of the Web service operation (Figure 11–8).

- **Number of Concurrent Threads** – The number of concurrent threads on which the invocations should be sent. The default is 5 threads.
- **Number of Loops per Thread** – The number of times to invoke the operation. The default is 10 times.
- **Delay in Milliseconds** – The number of milliseconds to wait between operation invocations. The default is 1000 milliseconds (1 second).

Figure 11–8 Stress Testing Parameters on the Test Page


Additional Test Options

Stress Test	☐ Enable
Concurrent Threads	5
Loops per Thread	10
Delay in Milliseconds	1000

When you invoke the test, a progress box indicates the test status.

When the test completes, a stress report page is returned. The report page identifies the service end point and operation being tested, the size of the message sent, the number of concurrent threads on which it is run, the number of times it is run on each thread, and the delay between each operation invocation.

Disabling the Test Page for a Web Service

Note: This section does not apply to JEE Web services.

Disabling the Test Page for a Web service allows you to increase security by reducing the externally visible details of an application that exposes Web services.

Note: Disabling the Test Enabled control affects only the Web service's externally-visible test page. It does not affect the Web service test feature described in this chapter.

To disable the Test Page using Fusion Middleware Control

1. Navigate to the Web Services Summary page, as described in "Navigating to the Web Services Summary Page for an Application" on page 6-4.
2. In the Web Service Details section of the page, click on the plus (+) for the Web service to display the Web service ports if they are not already displayed.
3. Click the name of the port to navigate to the Web Service Endpoints page.
4. Click the **Configuration** tab.
5. In the Test Enabled field, select **False** from the list.
6. Click **Apply**.

12 Monitoring the Performance of Web Services

This chapter describes how to monitor the performance of a Web service. The chapter includes the following sections:

- Overview of Performance Monitoring
- Viewing Web Service Statistics from the Summary Page
- Viewing Web Service Statistics for a Server Instance
- Viewing Web Service-Specific Statistics
- Viewing Endpoint-Specific Operations Statistics
- Viewing Policy Security Violations for an Endpoint

In addition to the monitoring features described in this chapter, see "Analyzing Policy Usage" on page 7-19 to analyze how policies are used by one or more Web services.

Overview of Performance Monitoring

Note: Not all of the monitoring features described in this chapter apply to Java EE Web services.

From the Web Services home page, you can do the following:

- Monitor Web services faults, including Security, Reliable Messaging, MTOM, Management, and Service faults.
- Monitor Security failures, including authentication, authorization, message integrity, and message confidentiality failures.
- Configure your Web services ports, including enabling and disabling the port, attaching policies to Web services, and enabling or disabling policies.

The Application home page also displays select Web service details if the application includes Web services.

When Are Web Service Statistics Started or Reset?

The statistics described in this chapter are started or reset when any one of the following events occur:

- When the application is being deployed for the first time.
- When the application is redeployed.

- If the application is already deployed, and the hosting server is restarted.

Viewing Web Service Statistics from the Summary Page

The Web Services summary page for an application displays the collective **Summary** and fault/violation information for all Web services in the application, as shown in Figure 12–1.

The **Charts** section shows a graphical view of all security faults for a Web service.

To navigate to the Web Service Summary page for a Web service

1. In the navigator pane, expand **Application Deployments** to show the application for which you want to monitor the Web service performance.

   Select the application.

2. Using Fusion Middleware Control, click **Web Services**.

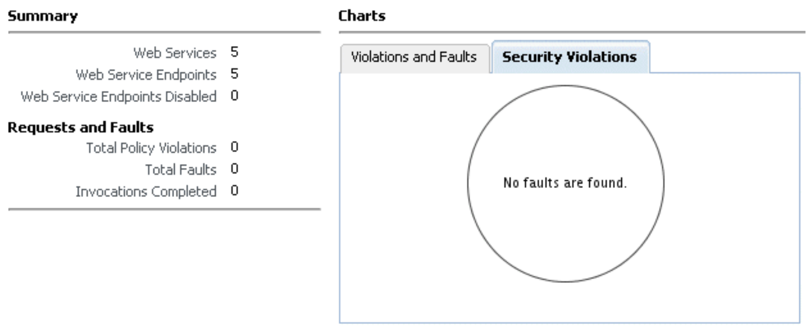
   The **Web Services Summary** page for this application is displayed.

   The page displays Web service endpoints as well as application-level metrics.

The following Web service-wide statistics are displayed:

- Web Services (Number of Web services in the application)
- Web Service Endpoints
- Web Service Endpoints Disabled
- Total Policy Violations
- Total Faults
- Invocations Completed

Figure 12–1 Web Services Performance Summary and Charts



Summary

Web Services	5
Web Service Endpoints	5
Web Service Endpoints Disabled	0

Requests and Faults

Total Policy Violations	0
Total Faults	0
Invocations Completed	0

Viewing Web Service Statistics for a Server Instance

The server-side Web services page displays statistics for all of the Web services on that server.

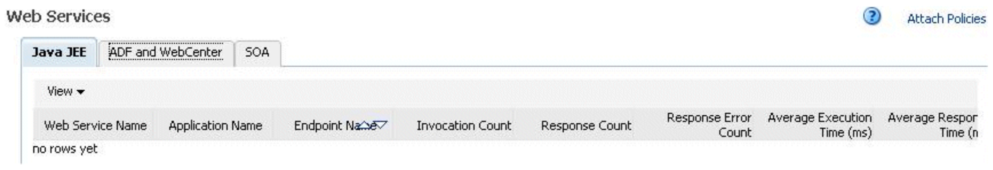
To view the Web service statistics for a server

1. In the navigator pane, expand **WebLogic Domain** to show the domain for which you want to see the policies. Select the domain.
2. Expand the domain to show the servers in that domain. Select the server for which you want to view the statistics.

3. Using Fusion Middleware Control, click **WebLogic Server**, and then **Web Services**.
4. The Web services statistics page for the server is displayed, as shown in Figure 12–2.

   Depending on what types of Web services you have deployed, tabs are available for the available Web service types: Java EE, ADF and Web Center, and SOA.

Figure 12–2 Web Services for a Server



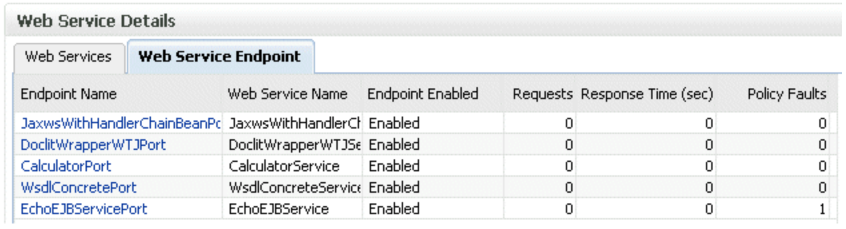
Web Service Name	Application Name	Endpoint Name	Invocation Count	Response Count	Response Error Count	Average Execution Time (ms)	Average Respor Time (n
no rows yet							

Viewing Web Service-Specific Statistics

The **Web Service Details** section of the Web Services Summary page displays statistics on a per-Web service basis, as shown in Figure 12–3. The following statistics are displayed:

- Endpoint Enabled
- Invocations Completed
- Response Time, in seconds
- Policy Violations
- Total Faults

Figure 12–3 Web Service-Specific Statistics



Endpoint Name	Web Service Name	Endpoint Enabled	Requests	Response Time (sec)	Policy Faults
JaxwsWithHandlerChainBeanPc	JaxwsWithHandlerCl	Enabled	0	0	0
DoclitWrapperWTJPort	DoclitWrapperWTJSe	Enabled	0	0	0
CalculatorPort	CalculatorService	Enabled	0	0	0
WsdlConcretePort	WsdlConcreteServic	Enabled	0	0	0
EchoEJBServicePort	EchoEJBService	Enabled	0	0	1

Viewing Endpoint-Specific Operations Statistics

To display operation statistics for a particular Web service endpoint, in the **Web Services Details** section of the **Web Service Summary** page select the endpoint for which you want to display the statistics.

The **Web Service Endpoint** page is displayed.

The following statistics are presented:

- Policy Reference Status
- Total Violations
- Security Violations

Viewing Policy Security Violations for an Endpoint

To display security violations for a particular Web service endpoint, do the following:

1. In the **Web Services Details** section of the **Web Service Summary** page select the endpoint for which you want to display the statistics.

   The **Web Service Endpoint Summary** page is displayed.

2. Click the **Policies** tab.

The following security violations are displayed:

- Total Violations
- Authentication violations
- Authorization violations
- Confidentiality violations
- Integrity

Part III

Advanced Administration

Note: For information about securing and administering WebLogic Web services, see Chapter 17, "Securing and Administering WebLogic Web Services."

Part III contains the following chapters:

- Chapter 13, "Advanced Administration"
- Chapter 14, "Creating Custom Assertions"
- Chapter 15, "Managing Horizontal Policy Migration"
- Chapter 16, "Diagnosing Problems"

13 Advanced Administration

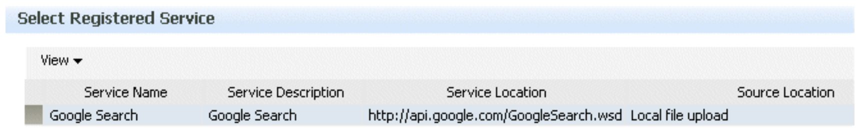
This chapter includes the following sections:

- Registering Web Services
- Auditing Web Services
- Managing the WSDL
- Managing Policy Assertion Templates
- About the Metadata Services (MDS) Repository
- Upgrading the Oracle WSM Policies in the MDS Repository
- Adding Security to a Running Client
- Managing Policy Accessor, Cache, and Interceptor Properties
- Setting Up the Java Object Cache
- Deleting the OracleSystemUser Default User

Registering Web Services

You can register a Web service so that you can later more conveniently reference the service from a selection list without having to specify a URL for a WSDL. For example, when testing a Web service, you can click the **Locate** icon and then select the WSDL from the registered services, as shown in Figure 13–1.

Figure 13–1 Selecting From a Registered Service



Service Name	Service Description	Service Location	Source Location
Google Search	Google Search	http://api.google.com/GoogleSearch.wsd	Local file upload

Fusion Middleware Control provides support for registering Web services that are published in WS-Inspection (WSIL) documents and UDDI v3 registries. Any service that is available in a WSIL document or a UDDI v3 registry can be registered.

When you register Web services, you do so by specifying any of the following:

- URL to a WSIL document
- File location of a WSIL document
- UDDI v3 URL

WSIL Basics

A key feature of the Web services model is the ability to make Web services widely available and discoverable. UDDI is one approach to publishing and discovery of Web services that centralizes information about businesses and their services in registries. Another emerging alternative standard is the Web Services Inspection Language (WSIL) specification.

WSIL defines an Extensible Markup Language (XML) format for referencing Web service descriptions. These references are contained in a WSIL document, and refer to Web service descriptions (for example, WSDL files) and to other aggregations of Web services (for example, another WSIL document or a UDDI registry).

WSIL documents are typically distributed by the Web service provider. These documents describe how to inspect the provider's Web site for available Web services. Therefore, the WSIL standard also defines rules for how WSIL documents should be made available to consumers of Web services.

The WSIL model decentralizes Web service discovery. In contrast to UDDI registries, which centralize information on multiple business entities and services, WSIL makes it possible to provide Web service description information from any location. Unlike UDDI, WSIL is not concerned about business entity information, and does not require a specific service description format. It assumes that you know who the service provider is and relies on other standards for Web service description, such as WSDL.

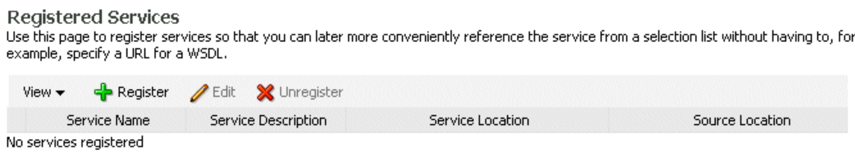
Registering a Web Service

Follow the steps in this section to register a service.

To register a service

1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to register a Web service.
2. Select the domain.
3. Using Fusion Middleware Control, click **WebLogic Domain** and then **Web Services** and then **Registered Services**. The Registered Service page appears, as shown in Figure 13–2.

Figure 13–2 Registering Services Page


Registered Services

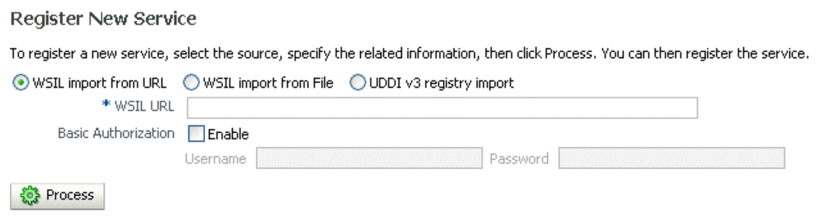
Use this page to register services so that you can later more conveniently reference the service from a selection list without having to, for example, specify a URL for a WSDL.

Service Name	Service Description	Service Location	Source Location

No services registered

4. Click **Register** to register a service. The Register New Service page appears, as shown in Figure 13–3.

Figure 13–3 Registering New Service Page


Register New Service

To register a new service, select the source, specify the related information, then click Process. You can then register the service.

[● WSIL import from URL] ○ WSIL import from File ○ UDDI v3 registry import

* WSIL URL	
Basic Authorization	☐ Enable
Username	Password

5. Select from **WSIL import from URL**, **WSIL import from File**, or **UDDI v3 registry import**.
6. For **WSIL import from URL**, enter the WSIL URL. Enable Basic Authorization and provide a username and password if required to access the WSIL.

   For **WSIL import from File**, enter the file specification and click **Browse**.

   For **UDDI v3 registry import**, enter the UDDI inquiry URL. For example, *http://somehost/uddi/inquiry*.
7. Click **Process** to parse your input.
8. For **UDDI v3 registry import**, the services available in the UDDI are displayed. Select one or more services to register.
9. Click **Register** to register the service or services.
10. If the registration is successful, the page expands to show the registered services. You can click **Edit** to change the service name and description from this page, if desired.
11. If the current WSIL also references other Web services, expand **References Available in WSIL** to display them. You can register the referenced Web services as well.

Viewing and Editing a Registered Web Service

Follow the steps in this section to view and edit a registered Web service.

1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to view the registered Web services.
2. Select the domain.
3. Using Fusion Middleware Control, click **WebLogic Domain** and then **Web Services** and then **Registered Services**. The Registered Service page appears, as shown in Figure 13–2.
4. The registered Web services are displayed. Select the Web service and click Edit to edit the registered service.

Unregistering a Web Service

Follow the steps in this section to unregister a Web service.

1. In the navigator pane, expand **WebLogic Domain** to show the domain in which you want to unregister a Web service.
2. Select the domain.

3. Using Fusion Middleware Control, click **WebLogic Domain** and then **Web Services** and then **Registered Services**. The Registered Service page appears, as shown in Figure 13–2.
4. The registered Web services are displayed. Select the Web service you want to unregister and click **Unregister**.

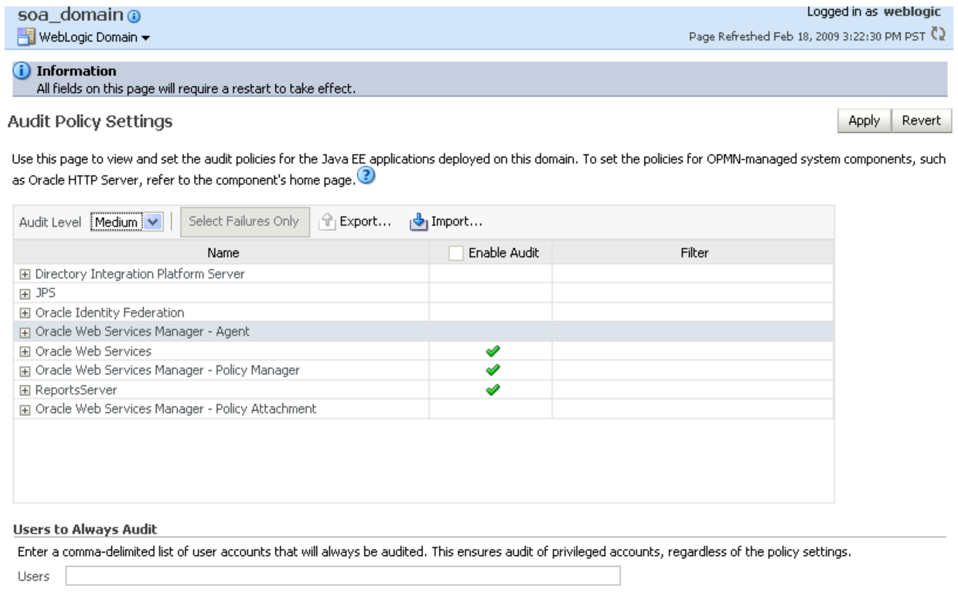
Auditing Web Services

Auditing describes the process of collecting and storing information about security events and the outcome of those events. An audit provides an electronic trail of selected system activity.

An audit *policy* defines the type and scope of events to be captured at runtime. Although a very large array of system and user events can occur during an operation, the events that are actually audited depend on the audit policies in effect at runtime. You can define component- or application-specific policies, or audit individual users.

You configure auditing for system components, including Web services, and applications at the domain level using the Audit Policy Settings page. You can audit SOA, ADF, and WebCenter services.

Figure 13–4 Audit Policy Settings Page



Information
All fields on this page will require a restart to take effect.

Audit Policy Settings

Use this page to view and set the audit policies for the Java EE applications deployed on this domain. To set the policies for OPMN-managed system components, such as Oracle HTTP Server, refer to the component's home page.

Name	Enable Audit	Filter
Directory Integration Platform Server		
JPS		
Oracle Identity Federation		
Oracle Web Services Manager - Agent		
Oracle Web Services	✓	
Oracle Web Services Manager - Policy Manager	✓	
ReportsServer	✓	
Oracle Web Services Manager - Policy Attachment		

Users to Always Audit

Enter a comma-delimited list of user accounts that will always be audited. This ensures audit of privileged accounts, regardless of the policy settings.

Users

The audit policies table, at the center of the page, displays the audits that are currently in effect. The table includes the following information:

- Name—Name of the system components and applications that you can audit.
- Enable Audit—Identifies the components and applications for which auditing is currently in effect.
- Filter—Specifies any filters that are currently in effect.

The following table summarizes the events that you can audit for Web services and the relevant component.

Table 13–1 Auditing Events for Web Services

Enable auditing for the following Web service events . . .	Using this system component . . .
■ User authentication. ■ User authorization. ■ Policy enforcement, including message integrity, message confidentiality, and security policy.	Oracle Web Services Manager—Agent
■ Web service requests sent and responses received. ■ SOAP faults incurred.	Oracle Web Services
■ Oracle WSM policy creation, deletion, or modification. ■ Assertion template creation, deletion, or modification.	Oracle Web Services Manager
■ Oracle WSM policy attachment.	Oracle Web Services Manager— Policy Attachment

You can also audit the events for a specific user, for example, you can audit all events by an administrator.

For more information about configuring audit policies, see "Configuring and Managing Auditing" in *Oracle Fusion Middleware Security Guide*.

The following sections describe how to define audit policies and view audit data:

- Configuring Audit Policies
- Managing Audit Data Collection and Storage
- Viewing Audit Reports

Configuring Audit Policies

To configure audit policies:

1. In the Navigator pane, expand **WebLogic Domain**.
2. Click the domain for which you want to manage assertion templates.
3. From the WebLogic Domain menu select **Security > Audit Policy Settings**.

   The Audit Policy Settings page is displayed.
4. Select and audit level from the Audit Level menu.

   Valid audit levels include:

   - None—Disables auditing.
   - Low—Audits a small scope of events. The subset of events is predefined individually for each component. For example, for a given component, Low may collect authentication and authorization events only.
   - Medium—Audits a medium scope of events (which is a superset of the events collected at the Low level). For example, for a given component, Medium may collect authentication, authorization, and policy authoring events.
   - Custom—Enables you to provide a custom auditing policy.

   You can view the components and applications that are selected for audit at each level in the audit policies list. For all audit levels other than Custom, the information in the audit policies list is greyed out, as you cannot customize other audit level settings.

5. If you selected the Custom audit level, perform one of the following steps:

   - Select the information that you want to audit by clicking the associated checkbox in the Enable Audit column.

     You can audit at the following levels of granularity: All events for a component, all events within a component event group, an individual event, or a specific outcome of an individual event (such as success or failure).

     At the event outcome level, you can specify an edit filter. Filters are rules-based expressions that you can define to control the events that are returned. For example, you might specify an Initiator as a filter for policy management operations to track when policies were created, modified, or deleted by a specific user. To define a filter for an outcome level, click the **Edit Filter** icon in the appropriate column, specify the filter attributes, and click **OK**. The filter definition appears in the Filter column.

     Deselect the checkbox for a component at a higher level to customize auditing for its subcomponents. You can select all components and applications by checking the checkbox adjacent to the column name.

   - To audit only failures for all system components and applications, **Select Failures Only**.

     If selected, all checkboxes in the Enable Audit column are cleared.

6. If required, enter a comma-separated list of users in the Always Audit Users text box.

   Specified users will always be audited, regardless of whether auditing is enabled or disabled, and at what level auditing is set.

7. Click **Apply.**

   To revert all changes made during the current session, click **Revert**.

Managing Audit Data Collection and Storage

To manage the data collection and storage of audit information, you need to perform the following tasks:

- Set up and manage an audit data repository.

  You can store records using one of two repository modes: file and database. It is recommended that you use the database repository mode. The Oracle Business Intelligence Publisher-based audit reports only work in the database repository mode.

- Set up audit event collection.

For more information, see "Managing Audit Data Collection and Storage" in *Oracle Fusion Middleware Security Guide*.

Viewing Audit Reports

For database repositories, data is exposed through pre-defined reports in Oracle Business Intelligence Publisher.

A number of predefined reports are available, such as: authentication and authorization history, Oracle WSM policy enforcement and management, and so on.

For details about generating and viewing audit reports using Oracle Business Intelligence Publisher, see "Using Audit Analysis and Reporting" in *Oracle Fusion Middleware Security Guide*.

For file-based repositories, you can view the bus-stop files using a text editor and create your own custom queries.

Managing the WSDL

In some cases, you might not want the Web service WSDL to be accessible to the public. You can enable or disable public access to the WSDL from the Web Service Endpoint page.

Note: In some cases, a Web service client needs to access a WSDL during invocation. If public access to the WSDL is disabled, the client will need to have a local copy of the WSDL.

To manage the WSDL:

1. Navigate to the Web Service endpoint configuration page, as described in "Configuring the Web Service Port" on page 6-10.
2. On the Configuration tab, set the WSDL Enabled field to **True** or **False** to enable of disable public access to your WSDL, respectively.
3. Click **Apply.**

Managing Policy Assertion Templates

The following sections describe how to create and manage policy assertion templates.

- Navigating to the Web Services Assertion Templates Page
- Viewing an Assertion Template
- Searching for an Assertion Template
- Creating an Assertion Template
- Exporting an Assertion Template
- Importing an Assertion Template
- Editing an Assertion Template
- Deleting an Assertion Template

Navigating to the Web Services Assertion Templates Page

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

To navigate to the Web Services Assertion Templates page:

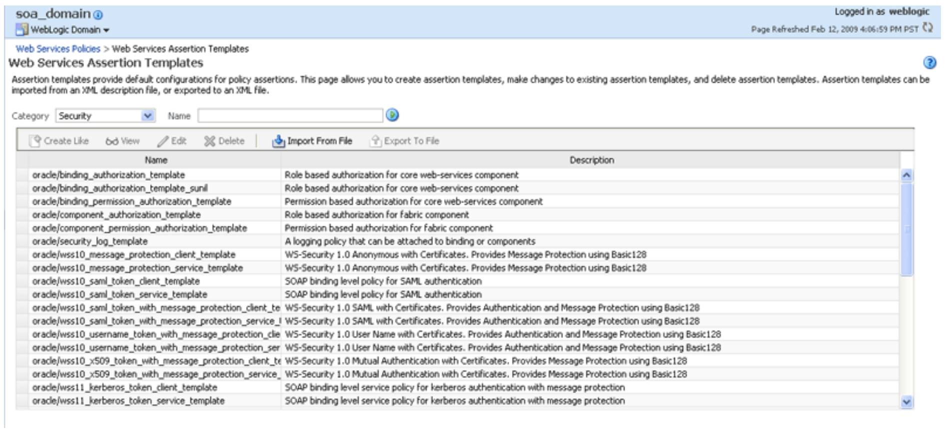
1. In the Navigator pane, expand **WebLogic Domain**.
2. Click the domain for which you want to manage assertion templates.
3. From the WebLogic Domain menu select **Web Services > Policies**.

   The Web Services Policies page is displayed.

4. Click **Web Services Assertion Templates** in the upper right corner of the page.

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

Figure 13–5 Web Services Assertion Templates Page



Web Services Assertion Templates

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

Category: Security   Name:

Name	Description
oracle/binding_authorization_template	Role based authorization for core web-services component
oracle/binding_authorization_template_sunil	Role based authorization for core web-services component
oracle/binding_permission_authorization_template	Permission based authorization for core web-services component
oracle/component_authorization_template	Role based authorization for fabric component
oracle/component_permission_authorization_template	Permission based authorization for fabric component
oracle/security_log_template	A logging policy that can be attached to binding or components
oracle/wss10_message_protection_client_template	WS-Security 1.0 Anonymous with Certificates. Provides Message Protection using Basic128
oracle/wss10_message_protection_service_template	WS-Security 1.0 Anonymous with Certificates. Provides Message Protection using Basic128
oracle/wss10_saml_token_client_template	SOAP binding level policy for SAML authentication
oracle/wss10_saml_token_service_template	SOAP binding level policy for SAML authentication
oracle/wss10_saml_token_with_message_protection_client_te	WS-Security 1.0 SAML with Certificates. Provides Authentication and Message Protection using Basic128
oracle/wss10_saml_token_with_message_protection_service_t	WS-Security 1.0 SAML with Certificates. Provides Authentication and Message Protection using Basic128
oracle/wss10_username_token_with_message_protection_clie	WS-Security 1.0 User Name with Certificates. Provides Authentication and Message Protection using Basic128
oracle/wss10_username_token_with_message_protection_ser	WS-Security 1.0 User Name with Certificates. Provides Authentication and Message Protection using Basic128
oracle/wss10_x509_token_with_message_protection_client_te	WS-Security 1.0 Mutual Authentication with Certificates. Provides Message Protection using Basic128
oracle/wss10_x509_token_with_message_protection_service_	WS-Security 1.0 Mutual Authentication with Certificates. Provides Message Protection using Basic128
oracle/wss11_kerberos_token_client_template	SOAP binding level service policy for kerberos authentication with message protection
oracle/wss11_kerberos_token_service_template	SOAP binding level service policy for kerberos authentication with message protection

Viewing an Assertion Template

To view an assertion template:

1. Navigate to the Web Services Assertion Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 13-7.
2. Select the assertion template from the Assertion Templates table that you want to view.
3. Click **View**.
4. In the View Template page, review the assertion.
5. When you are done, click **Return to Web Services Assertion Templates.**

Searching for an Assertion Template

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

To search for an asserting template:

1. Navigate to the Web Services Assertion Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 13-7.
2. Perform one or more of the following steps:
   - To search for assertion templates in a specific category (or all categories), select a category from the Category dropdown list.

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

     Specify any portion of the name of an assertion template to display all assertion templates that contain the string for the specified category.
3. Click **Go**.

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

Creating an Assertion Template

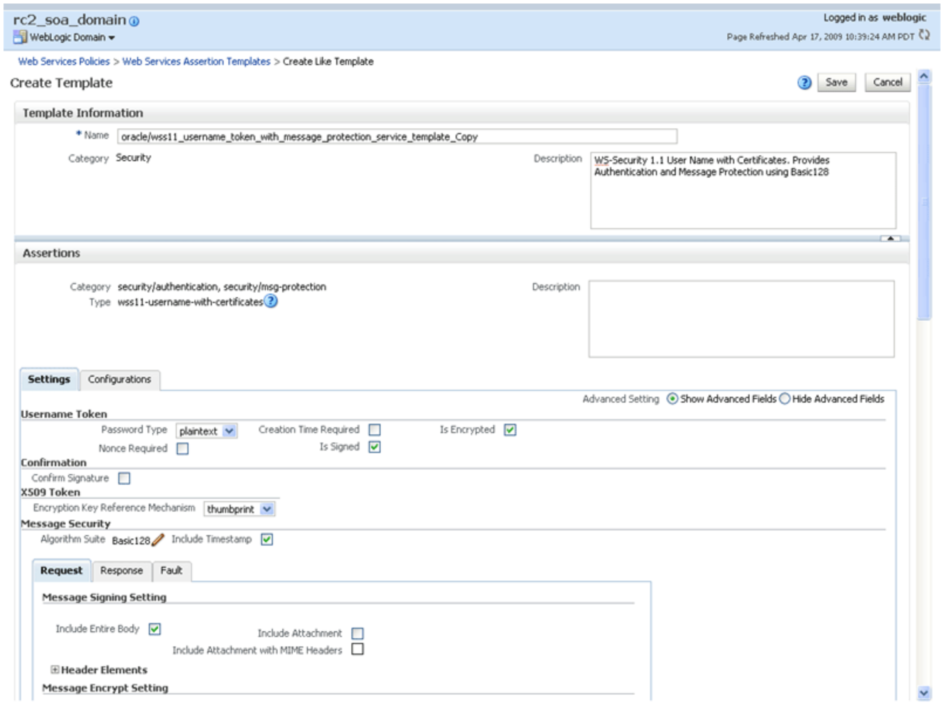
A new assertion template is created based on an existing assertion. Pick the assertion template that most closely matches the desired behavior, then make any changes required to get the desired behavior.

To create an assertion template:

1. Navigate to the Web Services Assertion Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 13-7.
2. Select the assertion template from the Assertion Templates table that you want to copy.
3. Click **Create Like**.

   The following shows the Create Template page.

Figure 13–6 Create Template Page



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

Web Services Policies > Web Services Assertion Templates > Create Like Template

Create Template

Template Information

* Name: oracle/wss11_username_token_with_message_protection_service_template_Copy
Category: Security
Description: WS-Security 1.1 User Name with Certificates. Provides Authentication and Message Protection using Basic128

Assertions

Category: security/authentication, security/msg-protection
Type: wss11-username-with-certificates
Description:

Advanced Setting: [✓ Show Advanced Fields] Hide Advanced Fields

Username Token
Password Type: plaintext
Creation Time Required: ☐
Is Encrypted: ✓
Nonce Required: ☐
Is Signed: ✓

Confirmation
Confirm Signature: ☐

X509 Token
Encryption Key Reference Mechanism: thumbprint

Message Security
Algorithm Suite: Basic128
Include Timestamp: ✓

Message Signing Setting
Include Entire Body: ✓
Include Attachment: ☐
Include Attachment with MIME Headers: ☐

Header Elements

Message Encrypt Setting

4. In the Copy Assertion Template box, edit the name of the assertion and enter a brief description.

   The word *Copy* is appended to the name of the copied assertion template and, by default, this is the name assigned to the new assertion template. For example, if the assertion template being copied is named *oracle/wss10_username_token_service_template*, then the default name of the copy is *oracle/wss10_username_token_service_template_Copy.*

   It is recommended that you change the name of this new assertion template to be more meaningful in your environment.
5. Click **OK.**

The assertion is added to the Assertion Templates table. You can now select the new assertion and click Edit to configure the assertion.

Exporting an Assertion Template

You can export individual assertion templates from Oracle Enterprise Manager Fusion Middleware Control. You can then copy the assertion template to a directory or import the assertion template to move it to another repository. Once moved, you can import the assertion template, as described in "Importing an Assertion Template" on page 13-10.

To export an assertion template:

1. Navigate to the Web Services Assertions Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 13-7.
2. Select the assertion template from the Assertion Templates table that you want to export to a file.
3. Click **Export to File**.

   You are prompted to open or save the file.
4. Select **Save File**.
5. Click **Ok**.
6. Navigate to the location on your local directory to which you want to save the file and update the filename as desired.
7. Click **Save**.

Importing an Assertion Template

To import an assertion template:

1. Navigate to the Web Services Assertions Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 13-7.
2. Click **Import From File**.

   You are prompted to provide the assertion template file.
3. Click **Browse** to navigate to the directory where the assertion template file is located and select the assertion template to be imported.
4. Click **OK**.

   The assertion template appears in the Assertion Templates table.

Editing an Assertion Template

To edit an assertion template:

1. Navigate to the Web Services Assertions Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 13-7.
2. Select the assertion template from the Assertion Templates table that you want to edit.
3. Click **Edit**.
4. Edit the assertion template as required.

5. Click **Save**.

Deleting an Assertion Template

To delete an assertion template:

1. Navigate to the Web Services Assertions Templates page, as described in "Navigating to the Web Services Assertion Templates Page" on page 13-7.
2. Select the assertion template from the Assertion Templates table that you want to delete.
3. Click **Delete**.

   You are prompted to confirm that you want to delete the assertion template.
4. Click **OK.**

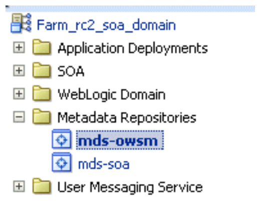
About the Metadata Services (MDS) Repository

When you install Oracle Fusion Middleware, you have the option of using a database-based MDS repository.

For a list of the databases that are supported for this release, see *Oracle Fusion Middleware Supported System Configurations* at:
`http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html`

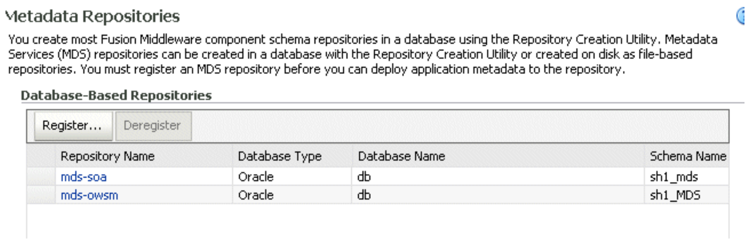
To register a MDS repository, expand Metadata Repositories in the Navigator pane, as shown in Figure 13–7.

Figure 13–7 Metadata Repository in Navigation Pane



Then, register a metadata repository, as shown in Figure 13–8.

Figure 13–8 Registering a Metadata 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

See "Managing the Oracle Metadata Repository" in the *Oracle Fusion Middleware Administrator's Guide* for information on managing the metadata repository.

Upgrading the Oracle WSM Policies in the MDS Repository

In Oracle WSM 11*g*, predefined and custom Oracle WSM policies are stored in the Oracle MDS repository. In subsequent releases, these predefined policies may be discontinued, changed, or additional predefined policies may be provided. You can use Oracle WebLogic Scripting Tool (WLST) commands to upgrade the repository with new or updated predefined policies when you install a new version of the Fusion Middleware software. You can also refresh the repository by deleting all Oracle WSM policies from the repository, including custom policies, and then repopulating it using the predefined policies provided in your installation. All of the policies in the repository are also revalidated when you upgrade the repository.

To upgrade the MDS repository:

1. Install the new version of Oracle Fusion Middleware.
2. Ensure that the MDS repository to be upgraded is registered with the new installation of Oracle Fusion Middleware. For more information, see "About the Metadata Services (MDS) Repository" on page 13-11.
3. Invoke WLST from the Oracle home in which you installed the new version of Oracle Fusion Middleware as described in "Accessing the Web Services Custom WLST Commands" on page 1-6.
4. Do one of the following:
   - To add new predefined policies that are provided in the latest installation of the Oracle Fusion Middleware software, enter the following command:

     ```
     upgradeWSMPolicyRepository()
     ```

     When you execute this command, a message is displayed indicating the policies that have been added to the repository. Note that existing predefined and user-defined custom policies in the repository are unchanged. The output message also displays a list of any existing predefined policies that have changed or discontinued in the latest release. If a policy has been discontinued and is no longer supported, Oracle recommends that you remove all references to it and then delete it using Oracle Enterprise Manager. If a predefined policy has changed in the latest release, Oracle recommends that you import the updated version of the policy using Oracle Enterprise Manager. For details about deleting and importing policies using Enterprise Manager, see Chapter 7, "Managing Web Service Policies."
   - To delete existing predefined policies stored in the repository and replace them with the latest set of predefined policies, use the following command:

     ```
     resetWSMPolicyRepository(false)
     ```

     User-defined custom policies are unchanged. An output message is displayed that lists the predefined policies that have been added, deleted, or replaced in the repository.
   - To delete all policies from the repository, including custom policies, and replace them with the latest set of predefined policies, use the following command:

     ```
     resetWSMPolicyRepository(true)
     ```

     Before you delete a policy, Oracle recommends that you verify that the policy is not attached to any policy subjects.

   When you execute any of these commands, all existing policies are revalidated.

For more information about these WLST commands, see "Policy Repository Upgrade Commands" in *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

Adding Security to a Running Client

Security policies can be attached to a running client using Oracle Enterprise Manager Fusion Middleware Control. You do not have to redeploy the client application in order to attach or detach policies from the client. See Chapter 8, "Attaching Policies to Web Services" for more information on how to attach policies using Fusion Middleware Control.

Managing Policy Accessor, Cache, and Interceptor Properties

You can manage properties for the following components from the Platform Policy Configuration page:

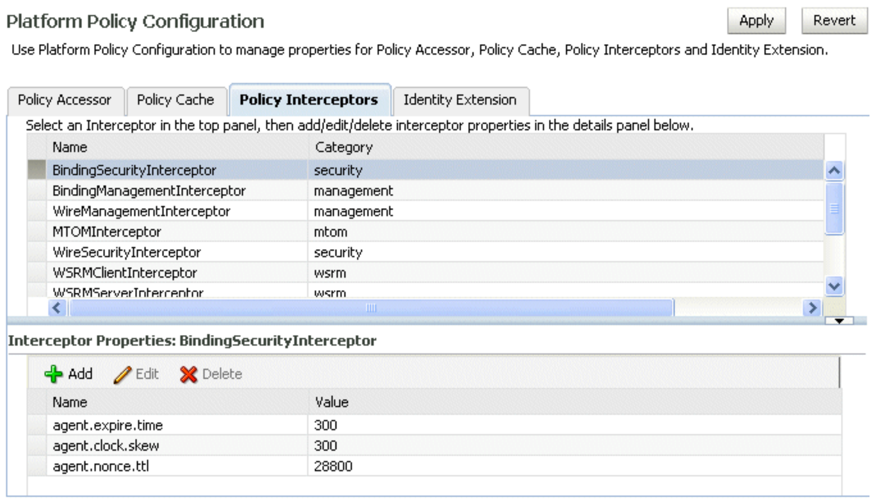
- Policy Accessor
- Policy Cache
- Policy Interceptors
- Identity Extension

To manage policy accessor, cache, interceptor, and identity extension properties:

1. In the navigator pane, expand **WebLogic Domain** to view the domains.
2. Select the domain for which you want to manage properties.
3. Select **WebLogic Domain> Web Services > Platform Policy Configuration**.

   The Platform Policy Configuration page appears, as shown in Figure 13–9.

Figure 13–9 Platform Policy Configuration Page



Use Platform Policy Configuration to manage properties for Policy Accessor, Policy Cache, Policy Interceptors and Identity Extension.

Select an Interceptor in the top panel, then add/edit/delete interceptor properties in the details panel below.

Name	Category
BindingSecurityInterceptor	security
BindingManagementInterceptor	management
WireManagementInterceptor	management
MTOMInterceptor	mtom
WireSecurityInterceptor	security
WSRMClientInterceptor	wsrm
WSRMServerInterceptor	wsrm

Interceptor Properties: BindingSecurityInterceptor

Name	Value
agent.expire.time	300
agent.clock.skew	300
agent.nonce.ttl	28800

4. Select the tab corresponding to the component for which you want to define properties: Policy Accessor, Policy Cache, Policy Interceptors, or Identity Extension.

5. If you selected the Policy Interceptors tab, select the interceptor for which you want to add properties in the list.
6. Perform one of the following tasks:
   - Click **Add** to define a new property.

     Enter the name of the property and value and click OK.
   - Select a property and click **Edit** to modify an existing property.
   - Select a property and click **Delete** to delete an existing property.
7. Click **Apply** to apply the property updates.

Setting Up the Java Object Cache

To protect against replay attacks, the wss_username_token_client_policy and wss_username_token_service_policy policies provide the option to require a nonce in the username token. A nonce is a unique number that can be used only once in a SOAP request and is used to prevent replay attacks.

The nonce is cached to prevent its reuse. However, in a cluster environment you must take steps to synchronize this cache across the managed servers. Otherwise, a request sent to a Web service running on one server can be replayed and sent to another managed server, where it will be processed. Oracle WSM uses an instance of Java Object Cache (JOC) to cache the nonce.

You use the *ORACLE_HOME/bin/configure-joc.py* Python script to configure the JOC on all of the managed servers in distributed mode. The script runs in WLST online mode and expects the Administration Server to be up and running.

Note: After configuring the Java Object Cache, restart all affected managed servers for the configurations to take effect.

Running the configure-joc.py Script

To enable the JOC in distributed mode, perform the following steps:

1. Connect to the Administration Server using the command-line Oracle WebLogic Scripting Tool (WLST), for example:

   ```
   ORACLE_HOME/oracle_common/common/bin/wlst.sh
   $ connect()
   ```

   Enter the Oracle WebLogic Administration user name and password when prompted.
2. After connecting to the Administration Server using WLST, start the script using the *execfile* command, for example:

   ```
   wls:/mydomain/serverConfig>execfile('ORACLE_HOME/bin/configure-joc.py')
   ```

3. Configure JOC for all the managed servers for a given cluster.

   Enter 'y' when the script prompts whether you want to specify a cluster name, and also specify the cluster name and discover port, when prompted. This discovers all the managed servers for the given cluster and configure the JOC f each managed server. The discover port is common for the entire JOC configuration across the cluster. For example:

   ```
   Do you want to specify a cluster name (y/n) <y>
   ```

```
Enter Cluster Name : Spaces_Cluster
Enter Discover Port : 9988
```

Here is a walkthrough for using *configure-joc.py* for HA environments:

```
execfile('ORACLE_HOME/bin/configure-joc.py')
.
Enter Hostnames (eg host1,host2) : SOAHOST1, SOAHOST2
.
Do you want to specify a cluster name (y/n) <y>y
.
Enter Cluster Name : Spaces_Cluster
.
Enter Discover Port : 9988
.
Enter Distribute Mode (true|false) <true> : true
.
Do you want to exclude any server(s) from JOC configuration (y/n) <n> n
```

The script can also be used to perform the following JOC configurations:

- Configure JOC for all specified managed servers.

  Enter 'n' when the script prompts whether you want to specify a cluster name, and also specify the managed server and discover port, when prompted. For example:

  ```
  Do you want to specify a cluster name (y/n) <y>n
  Enter Managed Server and Discover Port (eg WLS_Spaces1:9988, WLS_Spaces2:9988)
  : WLS_Spaces1:9988,WLS_Spaces2:9988
  ```
- Exclude JOC configuration for some managed servers.

  The script allows you to specify the list of managed servers for which the JOC configuration "DistributeMode" will be set to 'false'. Enter 'y' when the script prompts whether you want to exclude any servers from JOC configuration, and enter the managed server names to be excluded, when prompted. For example:

  ```
  Do you want to exclude any server(s) from JOC configuration (y/n) <n>y
  Exclude Managed Server List (eg Server1,Server2) : WLS_Spaces1,WLS_Spaces3
  ```

- Disable the distribution mode for all managed servers.

  The script allows you to disable the distribution to all the managed servers for a specified cluster. Specify 'false' when the script prompts for the distribution mode. By default, the distribution mode is set to 'true'.

Verify JOC configuration using the CacheWatcher utility. See *Oracle Fusion Middleware High Availability Guide*.

You can configure the Java Object Cache (JOC) using the HA Power Tools tab in the Oracle WebLogic Administration Console as described in the *Oracle Fusion Middleware High Availability Guide*.

Deleting the OracleSystemUser Default User

If you delete OracleSystemUser default user in favor of using the csf-key, then you must perform the following steps:

1. Add the following property and value to the Policy Accessor configuration. For more information, see "Managing Policy Accessor, Cache, and Interceptor Properties" on page 13-13.

   Name: jndi.lookup.csf.key

Value: basic.credentials

2. Ensure that the jndi.lookup.csf.key property is defined in the credential store. For example, basic.credential should be present in the credential store. For more information, see "Configuring the Credential Store Provider" on page 9-12.
3. Restart the server.

14 Creating Custom Assertions

This chapter describes how to create custom assertions. It includes the following sections:

- Overview of Custom Assertion Creation
- Step 1: Create the Custom Assertion Class
- Step 2: Create the Custom Policy File
- Step 3: Create the policy-config.xml File
- Step 4: Create the JAR File
- Step 5: Update Your CLASSPATH
- Step 6: Import the Custom Policy File
- Step 7: Attach the Custom Policy to a Web Service or Client

Overview of Custom Assertion Creation

If the predefined assertion templates, defined in "Predefined Assertion Templates" on page C-1, do not fit your needs, you can create your own custom assertions.

To create a custom assertion, you need to create the following files:

- Custom assertion class—Implements the Java class and its parsing and enforcement logic.
- Custom policy file—Enables you to define the bindings for and configure the custom assertion.
- policy-config.xml file—Registers the custom policy file.

You package the assertion class and policy-config.xml file as a JAR file and make the JAR file available in the CLASSPATH for your domain. Then, you import the custom policy file and attach it to your Web service or client, as required.

The following sections describe each step in the process.

Step 1: Create the Custom Assertion Class

Create the custom assertion class to execute and validate the logic of your policy assertion. The custom assertion class must extend `oracle.wsm.policyengine.impl.AssertionExecutor`.

When building the custom assertion class, ensure that the following JAR files are in your CLASSPATH: wsm-policy-core.jar and wsm-agent-core.jar.

The following example shows a custom assertion executor that can be used to validate the IP address of the request. If the IP address of the request is invalid, a `FAULT_FAILED_CHECK` exception is thrown.

For more information about the APIs that are available to you for developing your own custom assertion class, see the Java API Reference for Oracle Web Services Manager.

Example 14–1 Example Custom Assertion Class

```
package sampleassertion;

import oracle.wsm.common.sdk.IContext;
import oracle.wsm.common.sdk.IMessageContext;
import oracle.wsm.common.sdk.IResult;
import oracle.wsm.common.sdk.Result;
import oracle.wsm.common.sdk.WSMException;
import oracle.wsm.policy.model.IAssertionBindings;
import oracle.wsm.policy.model.IConfig;
import oracle.wsm.policy.model.IPropertySet;
import oracle.wsm.policy.model.ISimpleOracleAssertion;
import oracle.wsm.policy.model.impl.SimpleAssertion;
import oracle.wsm.policyengine.impl.AssertionExecutor;

public class IpAssertionExecutor extends AssertionExecutor {
    public IpAssertionExecutor() {
    }
    public void destroy() {
    }

    public void init(oracle.wsm.policy.model.IAssertion assertion,
                     oracle.wsm.policyengine.IExecutionContext econtext,
                     oracle.wsm.common.sdk.IContext context) {
        this.assertion = assertion;
        this.econtext = econtext;
    }
    public oracle.wsm.policyengine.IExecutionContext getExecutionContext() {
        return this.econtext;
    }
    public boolean isAssertionEnabled() {
        return ((ISimpleOracleAssertion)this.assertion).isEnforced();
    }
    public String getAssertionName() {
        return this.assertion.getQName().toString();
    }

    /**
     * @param context
     * @return
     */
    public IResult execute(IContext context) throws WSMException {
        try {
            IAssertionBindings bindings =
                ((SimpleAssertion)(this.assertion)).getBindings();
            IConfig config = bindings.getConfigs().get(0);
            IPropertySet propertyset = config.getPropertySets().get(0);
            String valid_ips =
                propertyset.getPropertyByName("valid_ips").getValue();
            String ipAddr = ((IMessageContext)context).getRemoteAddr();
            IResult result = new Result();
            if (valid_ips != null && valid_ips.trim().length() > 0) {
                String[] valid_ips_array = valid_ips.split(",");
                boolean isPresent = false;
                for (String valid_ip : valid_ips_array) {
                    if (ipAddr.equals(valid_ip.trim())) {
                        isPresent = true;
                    }
                }
                if (isPresent) {
                    result.setStatus(IResult.SUCCEEDED);
                } else {
                  result.setStatus(IResult.FAILED);
                  result.setFault(new WSMException(WSMException.FAULT_FAILED_CHECK));
                }
            } else {
                result.setStatus(IResult.SUCCEEDED);
            }
            return result;
        } catch (Exception e) {
            throw new WSMException(WSMException.FAULT_FAILED_CHECK, e);
        }
    }

    public oracle.wsm.common.sdk.IResult postExecute(oracle.wsm.common.sdk.IContext p1) {
        IResult result = new Result();
        result.setStatus(IResult.SUCCEEDED);
        return result;
    }
}
```

Step 2: Create the Custom Policy File

Create the custom policy file to define the bindings for and configure the custom assertion. "Schema Reference for Custom Assertions" on page E-1 describes the schema that you can use to construct your custom policy file and custom assertion.

The following example defines the oracle/ip_assertion_policy custom policy file. The assertion defines a comma-separated list of IP addresses that are valid for a request.

Example 14–2 Example Custom Policy File

```
<?xml version = '1.0' encoding = 'UTF-8'?>

<wsp:Policy xmlns="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
   orawsp:status="enabled"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" orawsp:category="security"
   orawsp:attachTo="binding.server" wsu:Id="ip_assertion_policy"
   xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
   wsp:Name="oracle/ip_assertion_policy">
      <orasp:ipAssertion orawsp:Silent="true" orawsp:Enforced="true"
         orawsp:name="WSSecurity IpAssertion Validator" orawsp:category="security/authentication">
            <orawsp:bindings>
                  <orawsp:Config orawsp:name="ipassertion" orawsp:configType="declarative">
                        <orawsp:PropertySet orawsp:name="valid_ips">
                              <orawsp:Property orawsp:name="valid_ips" orawsp:type="string"
                               orawsp:contentType="constant">
                                    <orawsp:Value>127.0.0.1,192.168.1.1</orawsp:Value>
                              </orawsp:Property>
                        </orawsp:PropertySet>
                   </orawsp:Config>
             </orawsp:bindings>
      </orasp:ipAssertion>
</wsp:Policy>
```

Step 3: Create the policy-config.xml File

Create a policy-config.xml file that defines an entry for the new assertion and associates it with its executor class.

The following defines the format for the policy-config.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<policy-config>
    <policy-model-config>
        <entry>
           <key namespace="namespace" elementName="elementname"/>
           <executor-classname>assertionclass</executor-classname>
        </entry>
    </policy-model-config>
</policy-config>
```

The following table lists the attributes for the key element.

Table 14–1 Attributes for Key Element

Attribute	Description
namespace	Namespace of the policy. This value must match the namespace defined in the custom policy file (in Step 2). In Example 14–2, the namespace is defined as part of the <wsp:Policy> tag as follows: `xmlns:orasp="`**`http://schemas.oracle.com/ws/2006/01/securitypolicy`**`"`
elementName	Name of the element. This value must match the assertion name defined in the custom policy file (in Step 2). In Example 14–2, the element name ipAssertion is defined in the following tag: `<orasp:`**`ipAssertion`**` orawsp:Silent="true" orawsp:Enforced="true" orawsp:name="WSSecurity IpAssertion Validator" orawsp:category="security/authentication">`

The following provides an example of a the policy-config.xml file with an entry for the ipAssertion policy.

Example 14–3 Example policy-config.xml File

```
<?xml version="1.0" encoding="UTF-8"?>
<policy-config>
    <policy-model-config>
        <entry>
            <key namespace="http://schemas.oracle.com/ws/2006/01/securitypolicy"
elementName="ipAssertion"/>
            <executor-classname>sampleassertion.IpAssertionExecutor</executor-classname>
        </entry>
    </policy-model-config>
</policy-config>
```

Step 4: Create the JAR File

Create the custom assertion JAR file that includes the IPAssertionExecutor class and the policy-config.xml file. You can use Oracle JDeveloper, other IDE, or the jar tool to generate the JAR file.

Step 5: Update Your CLASSPATH

You need to add the following files to your CLASSPATH:

- Custom assertion JAR file so that the custom assertion execution class is available in the server environment.
- wsm-policy-core.jar and wsm-agent-core.jar required for building the custom assertion class.

Add the custom assertion JAR to your CLASSPATH by performing the following steps:

1. Stop the WebLogic Server.

   For more information on stopping the WebLogic Server, see *Managing Server Startup and Shutdown for Oracle WebLogic Server*.

2. Copy the custom assertion JAR file created in Step 4 to the following directory: $DOMAIN_HOME/lib.
3. Restart the WebLogic Server.

   For more information on restarting the WebLogic Server, see *Managing Server Startup and Shutdown for Oracle WebLogic Server*.

Step 6: Import the Custom Policy File

Before you can attach the custom policy to a Web service, you must import it using the procedure described in "Importing Web Service Policies" on page 7-7.

Step 7: Attach the Custom Policy to a Web Service or Client

Attach the custom policy to a Web service using the steps described in "Attaching Policies to Web Services" on page 8-1.

15

Managing Horizontal Policy Migration

Policies can be migrated through the different stages of the application development and deployment cycles (such as, development to production).

This chapter includes the following sections:

- Overview of Horizontal Policy Migration
- Migrating Policies
- Migrating Policy Configuration
- Migrating Assertion Templates

Overview of Horizontal Policy Migration

The following steps describe a typical scenario of how you would create a policy and migrate the policy through the different stages of the application development and deployment cycles.

1. Use Oracle Enterprise Manager Fusion Middleware Control to create a policy.

   For more information, see "Creating Web Service Policies" on page 7-4.
2. Export the policy to a file.

   For more information, see "Migrating Policies" on page 15-2.
3. Copy the policy file to policy store location in the Oracle JDeveloper environment.
4. Create a Web service in Oracle JDeveloper and attach the policy to the Web service.

   For more information, see "Using Policies with Web Services" in the "Developing with Web Services" section of the JDeveloper online help.
5. Deploy the Web service to the staging server, and test the Web service.

   For more information, see "Developing Web Services" in the JDeveloper online help.
6. Import the policy to the production server environment.

   For more information, see "Migrating Policies" on page 15-2.
7. Migrate the following information, as required:
   - Policy configuration. See "Migrating Policy Configuration" on page 15-2.
   - Assertion templates. See "Migrating Assertion Templates" on page 15-5.
8. Deploy the application into the production environment, and test the Web service.

See "Deploying Web Services Applications" on page 5-1 and "Testing Web Services" on page 11-1.

Migrating Policies

You can export individual policies from Oracle Enterprise Manager Fusion Middleware Control. You can then copy the policy to a directory or import the policy to move it to another repository.

For details about exporting and importing policies, see the following section in "Managing Web Service Policies" on page 7-1:

- "Exporting Web Service Policies" on page 7-14
- "Importing Web Service Policies" on page 7-7

Alternatively, you can use the exportMetadata and importMetadata WLST commands to export and import the policies. The following describes the steps required:

To migrate policies using WLST commands:

1. Export the Oracle WSM policies to a local directory. For example, to export all Oracle WSM artifacts to the /exported/owsm_policies directory:

   ```
   exportMetadata(application='wsm-pm',server='<server_name>',
   docs='/policies/mycompany/**',toLocation='/exported/owsm_policies')
   ```

2. Move the files to the new machine. Ensure that the Oracle WSM Policy Manager is deployed on the new machine.
3. Import the Oracle WSM policies. For example, to import all Oracle WSM artifacts from the /toimport/owsm_policies directory:

   ```
   importMetadata(application='wsm-pm',server='<server_name>',
   fromLocation='/toimport/owsm_policies', docs='/policy/mycompany/**')
   ```

Note: Care should be taken when specifying the docs parameter. If the value /** is specified, then all objects are exported or imported, including policies, assertion templates, and policy attachments. Transferring policy attachments will introduce errors into the usage analysis numbers reported in the Fusion Middleware Control if the source and target environments are not identical. It is recommended that a more specific path be used whenever exporting and importing policies or assertion templates.

For more information about the WLST commands, see *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*.

Migrating Policy Configuration

The following sections describe how to migrate the configuration artifacts for Oracle WSM policies. Sections include:

- Migrating Keystores
- Migrating Users and Groups
- Migrating Credentials

- Migrating Oracle Platform Security Services Application and System Policies
- Migrating Oracle Platform Security Services Configuration
- Migrating SSL
- Migrating Kerberos Configuration

Migrating Keystores

If you are using message protection policies, you need to migrate your keystores. To migrate keystores:

1. Manually copy your keystores to the new environment.

   For Java SE applications, copy the keystore to a user-defined location. For Java EE applications, copy the keystore to the same directory as the jps-config.xml file, namely *DOMAIN_HOME*/config/fmwconfig.

2. By default, the keystore is named default-keystore.jks. If you have renamed the keystore, you must configure the keystore name in the Oracle Platform Security Services keystore service instance.

For information about configuring the keystore, see "Setting up the Keystore for Message Protection" on page 9-7.

Migrating Users and Groups

Users and groups are maintained as part of the WebLogic Server security realm.

To migrate users and groups in embedded LDAP, you can migrate the data using either the Oracle WebLogic Administration Console or WLST. For a complete description of the steps required, see "Migrating Security Data" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

To migrate users and groups in an LDAP store, there is no migration path. You need to recreate the users and groups and specify the assignments in the LDAP store in the new environment. See "Configuring Authentication Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

Migrating Credentials

There are two types of credentials maintained in the credential store that you may need to migrate:

- Username and password
- Keystore and encryption key passwords

The migration steps are described in the sections below.

Migrating Username and Password

If users are stored in an embedded LDAP and migrated, as described in "Migrating Users and Groups" on page 15-3, then you simply migrate the existing credentials to the new credential store. For a complete description of the steps required, see "Migrating Security Data" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

If users are stored in an LDAP store, there is no automated migration path. You need to recreate the credentials in the credential store. For more information about configuring credentials, see "Configuring the Credential Store Provider" on page 9-12.

Migrating Keystores and Encryption Key Passwords

You can migrate keystores and encryption key passwords manually using the procedure described in "Migrating Credentials Manually" in "Deploying Secure Applications" in *Oracle Fusion Middleware Security Guide*.

Migrating Oracle Platform Security Services Application and System Policies

If your Web service uses authorization policies, you must migrate the Oracle Platform Security Services application and system policies that grant permissions. For more information, see "Migrating Policies with the Command migrateSecurityStore" in "OPSS Authorization and the Policy Store" in *Oracle Fusion Middleware Security Guide*.

Migrating Oracle Platform Security Services Configuration

There is no automated migration path for Oracle Platform Security Services configuration. You must recreate the configuration in the new environment.

There are three types of configurations in the Oracle Platform Security Services that you may need to recreate:

- SAML trusted assertion issuer names (applicable for all SAML policies).

  If you use the default configuration for SAML trusted issuer configuration, then no migration is required. For information about configuring SAML in the new environment, see "Configuring the SAML and Kerberos Login Modules" on page 9-15.

- Keystore locations and CSF key configuration for keystore and keystore password (applicable for message protection policies only).

  If you use the default configuration for keystores, then no migration is required. For information about configuring keystores in the new environment, see "Setting up the Keystore for Message Protection" on page 9-7.

- Keytab location and service principal name (applicable to Kerberos policy).

  For information about configuring the keytab location and service principal name in the new environment, see "Configuring the SAML and Kerberos Login Modules" on page 9-15.

Migrating SSL

There is no automated migration path for SSL configuration. You must configure SSL keystores and settings in the new environment. For more information about configuring SSL keystores and settings in the new environment, see "Configuring Keystores for SSL" on page 9-1.

Migrating Kerberos Configuration

To migrate the Kerberos configuration:

1. Copy the Kerberos configuration file to the new environment, matching the directory structure. The Kerberos configuration file is located in the following locations, based on your operating system:

   - **UNIX**: /etc/krb5.conf
   - **Windows**: C:\windows\krb5.ini

2. Initialize the ticket cache with the correct credentials.

For more information, see "Using Kerberos Tokens" on page 9-19.

Migrating Assertion Templates

You can export individual assertion templates from Oracle Enterprise Manager Fusion Middleware Control. You can then copy the policy to a directory or import the policy to move it to another repository.

For details about exporting and importing assertion templates, see the following sections:

- "Exporting an Assertion Template" on page 13-10
- "Importing an Assertion Template" on page 13-10

16
Diagnosing Problems

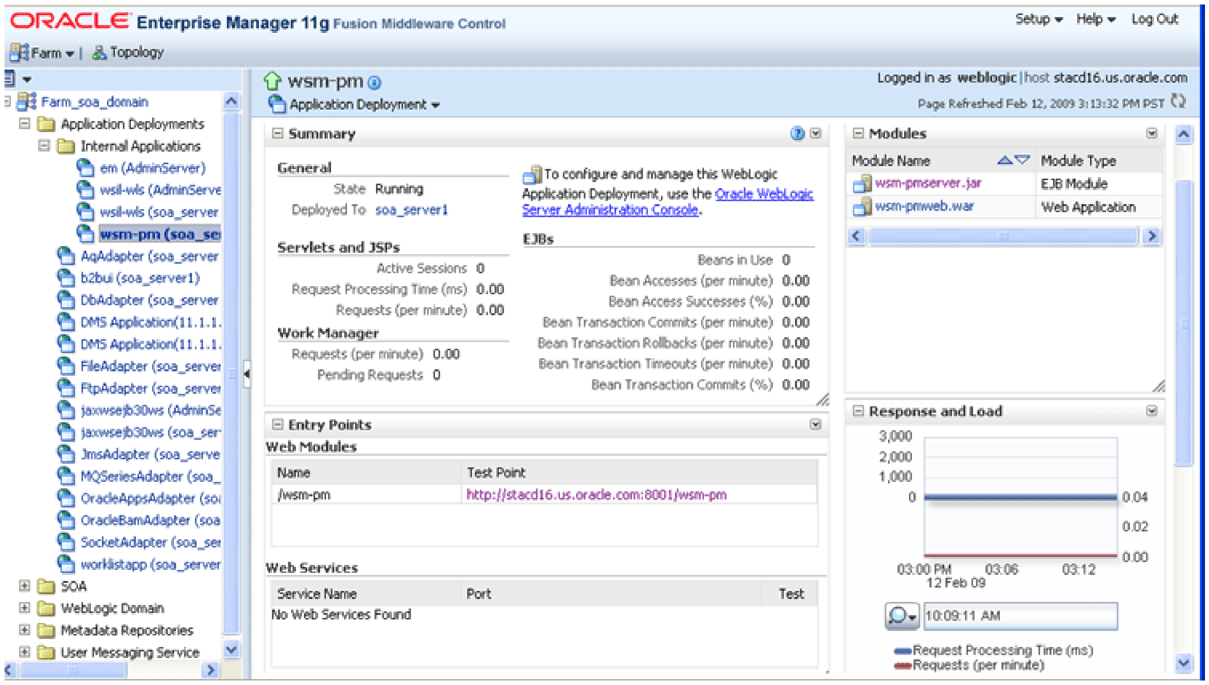
This chapter contains the following sections:

- Diagnosing Problems with Oracle WSM Policy Manager
- Diagnosing Problems Using Logs
- Configuring a Diagnostic Logger for a Web Service

Diagnosing Problems with Oracle WSM Policy Manager

The Oracle WSM Policy Manager manages all Oracle WSM policies and needs to be running in order to use the Oracle WSM policy framework. You can check the current state of the Policy Manager and review its response time, load, and other data from the Oracle WSM Policy Manager page, shown in the following figure.

Figure 16–1 Oracle WSM Policy Manager Page



General

State Running
Deployed To soa_server1

To configure and manage this WebLogic Application Deployment, use the Oracle WebLogic Server Administration Console.

Servlets and JSPs

Active Sessions 0
Request Processing Time (ms) 0.00
Requests (per minute) 0.00

Work Manager

Requests (per minute) 0.00
Pending Requests 0

EJBs

Beans in Use 0
Bean Accesses (per minute) 0.00
Bean Access Successes (%) 0.00
Bean Transaction Commits (per minute) 0.00
Bean Transaction Rollbacks (per minute) 0.00
Bean Transaction Timeouts (per minute) 0.00
Bean Transaction Commits (%) 0.00

Module Name	Module Type
wsm-pmserver.jar	EJB Module
wsm-pmweb.war	Web Application

Entry Points

Web Modules

Name	Test Point
/wsm-pm	http://stacd16.us.oracle.com:8001/wsm-pm

Web Services

Service Name	Port	Test
No Web Services Found		

To view the Oracle WSM Policy Manager page:

1. In the navigator pane, expand **Application Deployments**.
2. Expand **Internal Applications**.

3. Click **wsm-pm**.

   The Oracle WSM Policy Manager home page is displayed.

4. Perform one or more of the following tasks:

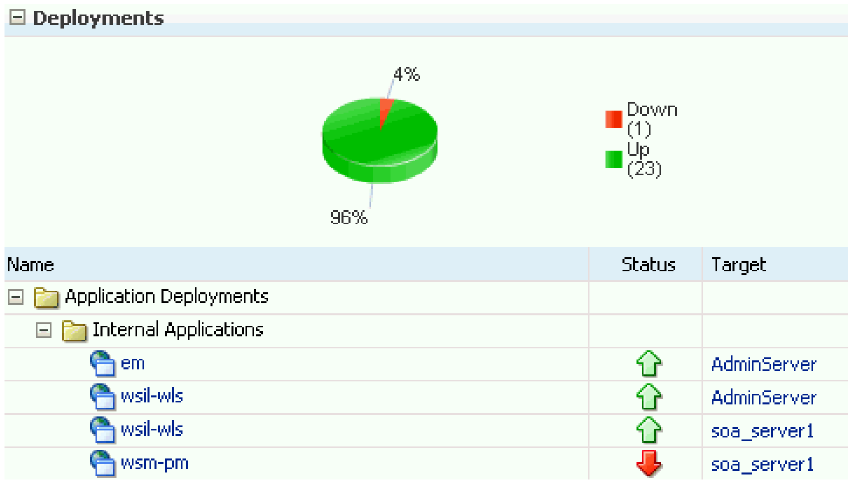
   - Check the current state of the Policy Manager and identify the server to which it is deployed.
   - View the response time and current load.
   - Click the Test Point URL to validate the Policy Manager. Click the Validate Policy Manager link. If operational, a list of the predefined policies is displayed with descriptions.

   Note: You can navigate to the Test Point URL by entering the following URL into a browser: `http://`*`host`*`:`*`port`*`/wsm-pm`.

The following lists the signs that indicate the Oracle WSM Policy Manager is not running. You can restart the wsm-pm application, as described in "Starting and Stopping Applications Using Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide*.

- The Policy Manager state is shown as being shutdown in the Oracle WSM Policy Manager home page, shown in Figure 16–1.
- The wsm-pm internal application deployment displays as being shutdown on the Farm page in the Enterprise Manager, as shown in the following figure.

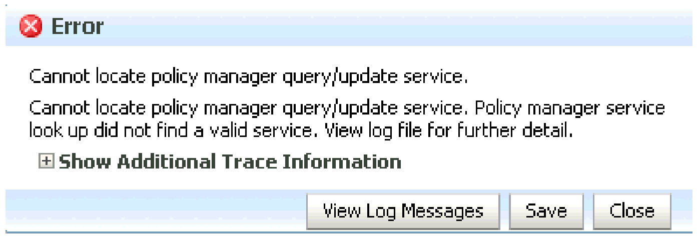
Figure 16–2 Oracle WSM Policy Manager Shutdown (Farm Page)



Name	Status	Target
Application Deployments		
Internal Applications		
em	↑	AdminServer
wsil-wls	↑	AdminServer
wsil-wls	↑	soa_server1
wsm-pm	↓	soa_server1

- An error dialog box similar to the following displays when you attempt to access the Oracle WSM policy management pages in the Enterprise Manager. This error information is also written to the diagnostic log file, as described in "Reviewing Sample Logs" on page 16-7.

Figure 16–3 Error Message—Oracle WSM Policy Manager Unavailable


Error

Cannot locate policy manager query/update service.

Cannot locate policy manager query/update service. Policy manager service look up did not find a valid service. View log file for further detail.

Diagnosing Problems Using Logs

Oracle Fusion Middleware components, including Web services, generate log files containing messages that record all types of events, including startup and shutdown information, errors, warning messages, access information on HTTP requests, and so on. Each log message includes specific information such as time, component ID, and user to assist you in pinpointing and diagnosing problems that arise.

You can review log messages to diagnose problems with specific components, such as Web services. There are two categories of log files that you can reference to assist in diagnosing problems with Web services:

- **Diagnostic logs**—Enable you to access diagnostic data about specific feature components in Oracle Fusion Middleware. For more information, see "Using Diagnostic Logs for Web Services" on page 16-3.

  There is a set of predefined diagnostic loggers. You can configure your own diagnostic logger, as described in "Configuring a Diagnostic Logger for a Web Service" on page 16-9.

- **Message logs**—Enable you to view elements of the SOAP message request. You control message log creation using policies. For more information, see "Using Message Logs for Web Services" on page 16-6.

For more information about logging in Oracle Fusion Middleware, see "Managing Log Files and Diagnostic Data" in *Oracle Fusion Middleware Administrator's Guide*.

The following sections describe how to use diagnostic and message logs to diagnose problems. A set of sample logs is provided at the end of this section.

Using Diagnostic Logs for Web Services

Diagnostic logs enable you to access diagnostic data about specific feature components in Oracle Fusion Middleware.

The following sections describe how to view and manage diagnostic log files:

- Setting the Log Level for Diagnostic Logs
- Viewing Diagnostic Logs
- Filtering Diagnostic Logs

Setting the Log Level for Diagnostic Logs

You set the logging level for Web service and Oracle WSM components at the WebLogic Server level, using the Log Configuration page.

In addition, you can override the log levels set at the server level for a specific Web service endpoint from the Web service endpoint page. The logging level set at the Web service endpoint level must be "finer grained" than the level set at the WebLogic Server level. Otherwise, the logging level set at the WebLogic Server level will be used.

The following procedures describe how to set the log level for diagnostic logs at the WebLogic Server and Web service endpoint levels. For more information, see "Setting the Level of Information Written to Log Files" in *Oracle Fusion Middleware Administrator's Guide*.

To set the log level for diagnostics logs at the WebLogic Server level:

1. Navigate to the WebLogic Server for which you want to configure diagnostic logs.
   a. In the navigator pane, expand **WebLogic Domain**.
   b. Expand the domain.
   c. Select the desired server from the list.

      The WebLogic Server home page is displayed.
2. From the **WebLogic Sever** menu, select **Logs > Log Configuration**.

   The Log Configuration page is displayed.
3. Select the **Log Levels** tab.
4. Expand **Root Logger**.
5. Expand **oracle**.
6. Set the logging level for one or more of the following components:
   - oracle.webservices—Web service components.
   - oracle.wsm—Oracle WSM components.

   You can fine tune the logging level by expanding either of the above components and specifying the logging level at the subcomponent level. By default, the logging levels are inherited from the parent and set to NOTIFICATION: 1 (INFO) for the Web service and Oracle WSM components and subcomponents.

To set the log level for diagnostic logs at the Web service endpoint level:

1. Navigate to the Web service endpoint page, as described in "Viewing the Details for a Web Service Port" on page 6-5.
2. Click the **Configuration** tab.
3. Set the **Logging Level** field to one of the following settings: Severe, Warning, Information, Configuration, Fine, Finer, Finest or NULL.

Note: You can also set the log level at the Web service endpoint using the `setWebServiceConfiguration` WLST command. Set the `loggingLevel` property of the `itemProperties` argument to one of the following settings: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, or NULL. For details about using this command, see "Using WLST" in "Configuring the Web Service Port" on page 6-10.

Viewing Diagnostic Logs

You can view the diagnostic log files for an ADF and WebCenter Web service endpoint from the Log Messages page.

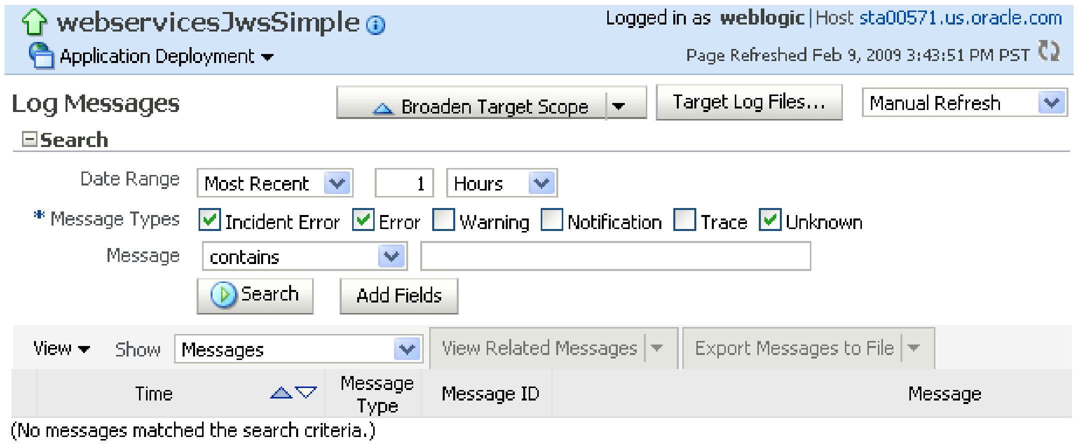
To view diagnostic logs for a Web service endpoint:

Navigate to the Web service endpoint page, as described in "Viewing the Details for a Web Service Port" on page 6-5, and in the Quick Links section of the Web Services Endpoint page (top right), click **Diagnostic Logs.**

Note: You can view a summary of all faults incurred by the Web services in your application. For more information, see "Monitoring the Performance of Web Services" on page 12-1.

The Log Messages page is displayed, as shown in the following figure.

Figure 16–4 Log Messages Page



(No messages matched the search criteria.)

Click on a message in the message area to view more details at the bottom of the page. If desired, you can export a message to a text, XML, or CSV file by selecting the messages on the list and clicking **Export Messages to File**.

You can control the message content displayed using the following controls:

- **Search**—Modify the search criteria. For more information, see "Filtering Diagnostic Logs" on page 16-6.
- **View menu**—Select the columns to display in the table. Click on a particular column to sort contents up or down.
- **Show menu**—Group messages by type or ID, or view them in chronological order.
- **View Related Messages**—View messages related to those selected on the list.
- **Broaden Target Scope**—Broaden the scope of messages displayed. You can broaden the scope to include all messages for the domain, WebLogic Server, or Farm.
- **Refresh menu**—Specify an automatic or manual refresh rate.

To view the contents of a generated log file:

- Click the log file icon associated with a message to view the contents of that log file.
- Click **Target Log Files...** to display the Log Files page and view or download the contents of all generated log files.

For more information, see "Searching and Viewing Log Files" in *Oracle Fusion Middleware Administrator's Guide*.

Filtering Diagnostic Logs

By default, the Log Messages page displays a summary of diagnostic messages logged over the last hour.

To filter diagnostic logs:

1. Filter the messages that are displayed by updating the search criteria using the following fields:
   - **Date Range**—Set the date range to one of the following:
     - Most Recent—Set the amount of time to define the duration.
     - Time Interval—Set the start and end dates to define the interval.
   - **Message Types**—Select the message types that you want to display.
   - **Add Fields**—Add other message fields to your search criteria, such as Message ID, Component, and so on.
2. Click **Search** once you have set the fields, as desired.

   The messages area is updated with the filtered results.

For more information, see "Searching and Viewing Log Files" in *Oracle Fusion Middleware Administrator's Guide*.

Using Message Logs for Web Services

Message logs enable you to access the contents of the SOAP message requests and responses for ADF and WebCenter Web services and clients. Messages logs are stored in a log file separate from the diagnostic messages, by default.

The following sections describe how to view and manage message log files:

- Configuring Message Logs
- Viewing Message Logs
- Filtering Message Logs

Configuring Message Logs

You configure message logs for a Web service or client by attaching a policy that contains a logging assertion.

There is one predefined logging assertion template: oracle/log_template. This template is configured to log the entire SOAP message for the Web service request and response. By default, all predefined Web service security policies use this logging assertion to capture the entire SOAP message before and after the primary security assertion is executed. By default, the log assertion is not enforced. You must enable it in order for the SOAP message to be logged in message logs. It is recommended that the logging assertion be enabled for debugging and auditing purposes only. For more information about the predefined logging policy, see "oracle/log_policy" on page B-23.

You can create your own logging policy or assertion template to further refine the elements of the SOAP message that are logged for the Web service request and response. For example, you may wish to view only the SOAP body of the request message. To create a new policy, following the procedure described in "Creating Web Service Policies" on page 7-4. You may wish to create a copy of the oracle/log_ template assertion template and configure it for use in the new policy. For more information about creating a new assertion template, see "Creating an Assertion Template" on page 13-9.

Viewing Message Logs

You can view the message log files for an ADF and WebCenter Web service endpoint from the Log Messages page.

To view message logs for a Web service endpoint:

Navigate to the Web service endpoint page, as described in "Viewing the Details for a Web Service Port" on page 6-5, and in the Quick Links section of the Web Services Endpoint page (top right), click **Message Logs.**

The Log Messages page is displayed, similar to Figure 16–4. For more details about the contents of the Log Messages page, see "Viewing Diagnostic Logs" on page 16-4.

Filtering Message Logs

By default, the Log Messages page displays a summary of SOAP messages logged over the last hour. You can filter the messages that are displayed by updating the search criteria. The process is the same as for diagnostic logs; for more information, see "Filtering Diagnostic Logs" on page 16-6.

By default, the Component and Module message fields are included as part of the Search criteria for message logs. The Component field is set to the WebLogic Server name; the Module field is set to oracle.wsm.msg.logging, which is the name of the message logging component.

Reviewing Sample Logs

The following sections provide excerpts from sample logs, demonstrating how to diagnose specific problems using the log entries.

- Sample Log: Oracle WSM Policy Manager Not Available
- Sample Log: Security Keystore Not Configured
- Sample Log: Certificate Not Available

Sample Log: Oracle WSM Policy Manager Not Available

The following sample log excerpt indicates that the Oracle WSM Policy Manager is down. To resolve this issue, restart the wsm-pm application, as described in "Starting and Stopping Applications Using Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide*.

```
2009-02-16 16:21:28,029 [[ACTIVE] ExecuteThread: '4' for queue:
 'weblogic.kernel.Default (self-tuning)']
 ERROR policymgr.PolicyManagerModelBean logp.251 -
 Service lookup failed with URL:t3://stadk13.us.oracle.com:7001/wsm-pm
 oracle.wsm.policymanager.PolicyManagerException: WSM-02118 :
 The query service cannot be created.
...
```

Sample Log: Security Keystore Not Configured

The following sample log excerpt indicates that an Oracle WSM security policy with message protection was applied, but the keystore was not configured. To resolve this security fault, configure the keystore, as described in "Setting up the Keystore for Message Protection" on page 9-7.

```
Feb 16, 2009 5:29:56 PM oracle.wsm.common.logging.WsmMessageLogger logSevere
 SEVERE: The specified Keystore file /scratch/sbollapa/stage131/user_
projects/domains/sai131_domain/config/fmwconfig/default-keystore.jks
 cannot be found; it either does not exist or its path is not included in the
 application classpath.
 Feb 16, 2009 5:29:56 PM oracle.wsm.common.logging.WsmMessageLogger logSevere
 SEVERE: Keystore is not properly configured in jps config.
 Feb 16, 2009 5:29:56 PM oracle.wsm.common.logging.WsmLogUtil log
 SEVERE: failure in OWSM Agent processRequest, category=security,
 function=agent.function.client, application=default, composite=pe3test3,
 modelObj=Service1, + policy=null, policyVersion=null, assertionName=null
 oracle.wsm.common.sdk.WSMException: WSM-00101 : The specified Keystore file
 /scratch/sbollapa/stage131/user_projects/domains/sai131_
domain/config/fmwconfig/default-keystore.jks cannot be found;
 it either does not exist or its path is not included in the application
classpath.
...
```

Sample Log: Certificate Not Available

The following sample log excerpt indicates that an Oracle WSM security policy with message protection was applied that required a security certificate that was not available in the keystore. To resolve this security fault, configure the keystore with a certificate, as described in "Setting up the Keystore for Message Protection" on page 9-7.

```
[2009-04-15T04:07:02.821-07:00] [jrfServer] [ERROR] [WSM-000062]
 [oracle.wsm.resources.security] [tid: [ACTIVE].ExecuteThread: '0' for
queue: 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>]
 [ecid: 0000I2dTFG7DScT6uBe9UH19tRyv000000,0:1] [WEBSERVICE_PORT.name:
 NonCAAsCAMessageProtectionPolicyPort] [APP: jaxwsservices]
 [J2EE_MODULE.name: NonCAAsCAMessageProtectionPolicy] [WEBSERVICE.name:
 NonCAAsCAMessageProtectionPolicyService] [J2EE_APP.name: jaxwsservices]
 [arg: oracle.wsm.security.SecurityException: WSM-00062 :
 The path to the certificate used for the signature is invalid.]

[2009-04-15T04:07:02.810-07:00] [jrfServer] [NOTIFICATION] []
 [oracle.wsm.security.policy.scenario.processor.Wss11X509TokenProcessor]
 [tid: [ACTIVE].ExecuteThread: '0' for queue: 'weblogic.kernel.Default
 (self-tuning)'] [userId: <anonymous>]
[ecid: 0000I2dTFG7DScT6uBe9UH19tRyv000000,0:1]
[WEBSERVICE_PORT.name: NonCAAsCAMessageProtectionPolicyPort]
[APP: jaxwsservices] [J2EE_MODULE.name: NonCAAsCAMessageProtectionPolicy]
 [WEBSERVICE.name: NonCAAsCAMessageProtectionPolicyService] [J2EE_APP.name:
 jaxwsservices] Certificate path validation failed for signing certificate

[2009-04-15T04:07:02.821-07:00] [jrfServer] [ERROR] [WSM-00006]
 [oracle.wsm.resources.security] [tid: [ACTIVE].ExecuteThread: '0' for queue:
 'weblogic.kernel.Default (self-tuning)'] [userId: <anonymous>]
[ecid: 0000I2dTFG7DScT6uBe9UH19tRyv000000,0:1] [WEBSERVICE_PORT.name:
 NonCAAsCAMessageProtectionPolicyPort] [APP: jaxwsservices]
[J2EE_MODULE.name: NonCAAsCAMessageProtectionPolicy] [WEBSERVICE.name:
 NonCAAsCAMessageProtectionPolicyService] [J2EE_APP.name: jaxwsservices]
[arg: oracle.wsm.security.SecurityException: WSM-00062 : The path to the
 certificate used for the signature is invalid.] Error in receiving the request:
 oracle.wsm.security.SecurityException: WSM-00062 : The path to the certificate
 used for the signature is invalid.
```

Configuring a Diagnostic Logger for a Web Service

To further organize your diagnostic data, you can configure a new diagnostic logger for a Web service. You can configure diagnostic loggers for SOA, ADF, and Web Center services.

By default, the following loggers are defined:

- odl-handler—Logs general diagnostic data for the Java EE components in the server.
- owsm-message-handler—Logs SOAP messages as per Oracle WSM logging policies.
- owsm-odl-handler—Logs diagnostic data for Oracle WSM components.

To configure a diagnostic logger for a Web service:

1. Navigate to the WebLogic Server for which you want to configure a diagnostic logger.
   a. In the navigator pane, expand **WebLogic Domain**.
   b. Expand the domain.
   c. Select the desired server from the list.

      The WebLogic Server home page is displayed.
2. From the **WebLogic Sever** menu, select **Logs > Log Configuration**.

   The Log Configuration page is displayed.
3. Select the **Log Files** tab.

   The current list of diagnostic loggers is displayed.
4. Click **Create**.

   Note: To copy the configuration for an existing diagnostic logger, select the logger and click **Create Like.**

   The Create Log File page is displayed.
5. Enter the data for the diagnostic logger, as follows.

Table 16–1 Fields in Create Log File Page

Field	Description
Handler Class	Handler class. Leave this value set to oracle.core.ojdl.looging.ODLHandlerFactory.
Log File	Name of the log file.
Log Path	Path to the log file.
Log File Format	Format of the log file. Valid values are text or XML.

Table 16–1 (Cont.) Fields in Create Log File Page

Field	Description
Log Level	Default log level for the diagnostic logger. Select a log level from the list. Valid values include: INCIDENT_ERROR ■ INCIDENT_ERROR:1 (SEVERE+100) ■ ERROR:1 (SEVERE) ■ WARNING:1 (WARNING) ■ NOTIFICATION:1 (INFO) ■ NOTIFICATION:16 (CONFIG) ■ TRACE:1 (FINE) ■ TRACE:16 (FINER) ■ TRACE:32 (FINEST)
Use Default Attributes	Flag that specifies whether to use default attributes for the diagnostic logger.
Supplemental Attributes	Supplemental attributes required.
Loggers to Associate	Components to associate with the logger.
Rotation Policy	Specify whether you wish to rotate log files based on file size of length of time. For more information, see "Configuring Log File Rotation" in *Oracle Fusion Middleware Administrator's Guide*.

6. Click **OK** to create the diagnostic logger.

Part IV

WebLogic Web Service Administration

Part IV contains the following chapter:

- Chapter 17, "Securing and Administering WebLogic Web Services"

17

Securing and Administering WebLogic Web Services

This chapter describes how to secure and administer WebLogic Web services, including the following sections:

- Steps to Secure and Administer WebLogic Web Services
- Attaching Policies to WebLogic Web Services and Clients

Steps to Secure and Administer WebLogic Web Services

Table 17–1 summarizes the steps required to administer and secure WebLogic Web services. For information about developing WebLogic Web services, see *Getting Started With JAX-WS Web Services for Oracle WebLogic Server*.

Table 17–1 Steps to Administer and Secure WebLogic Web Services

#	Step	Description
1	Deploy and administer the WebLogic Web service.	Use the Oracle WebLogic Server Administration Console to perform the following deployment and administration tasks: ■ Deploy a WebLogic Web service and view deployed services. ■ Start and stop a WebLogic Web service. ■ View the WebLogic Web service configuration. ■ Delete a WebLogic Web service. ■ View the SOAP message handlers. ■ View the WSDL. For more information, see "Web Services" in the *Oracle WebLogic Server Administration Console Online Help*.
2	Attach the security and management policies to your WebLogic Web services and clients.	You can attach two types of policies to WebLogic Web services and clients at design and deployment time: Oracle WSM and WebLogic Web service policies. For details, see "Attaching Policies to WebLogic Web Services and Clients" on page 17-2.
3	Test the WebLogic Web services.	See "Testing Web Services" on page 11-1.
4	Monitor the performance of WebLogic Web services.	See "Monitoring the Performance of Web Services" on page 12-1.

Attaching Policies to WebLogic Web Services and Clients

In Oracle Fusion Middleware 11*g* Release 1 (11.1.1), you can provide security and management policy enforcement of WebLogic Web services using one of the following policy types: *Oracle WSM* or *WebLogic Web service*.

The following table describes each policy type.

Table 17–2 Policy Types Supported by WebLogic Web Services

Type	Description
Oracle Web Services Manager (WSM) Policy	Provided by the Oracle WSM. For more information about the Oracle WSM and the predefined policies, see "Understanding Oracle WSM Policy Framework" on page 3-1. You can attach Oracle WSM policies to WebLogic JAX-WS Web services only.
WebLogic Web Service Policy	Provided by Oracle WebLogic Server. For more information about the WebLogic Web service policies, see *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server*. A subset of WebLogic Web service policies interoperate with Oracle WSM policies. For more information, see "Interoperability with Oracle WebLogic Server 11g Web Service Security Environments" in *Interoperability Guide for Oracle Web Services Manager*.

Note: It is recommended that you use Oracle WSM policies whenever possible. You cannot mix your use of Oracle WSM and WebLogic Web service policies.

The following sections describe how to attach each type of policy to WebLogic Web services and clients.

- Attaching Oracle WSM Policies to WebLogic Web Services
- Attaching Oracle WSM Policies to WebLogic Web Service Clients
- Attaching WebLogic Web Service Policies to WebLogic Web Services
- Attaching WebLogic Web Service Policies to WebLogic Web Service Clients

Attaching Oracle WSM Policies to WebLogic Web Services

You attach Oracle WSM policies to WebLogic Web services at design time and after the Web service has been deployed.

- At design time, use the `@SecurityPolicy` and `@SecurityPolicies` JWS annotations in your JWS file to associate policy files with your Web service. You can associate any number of policy files with a Web service, although it is up to you to ensure that the assertions do not contradict each other. You can specify a policy file at the class level of your JWS file. For more information, see the following sections:
  - "Using Oracle Web Service Security Policies" in *Securing WebLogic Web Services for Oracle WebLogic Server*.
  - "Using Policies with Web Services" in "Developing with Web Services" in the Oracle JDeveloper online help.
- After the Web service has been deployed, use the Oracle WebLogic Server Administration Console to attach Oracle WSM policies to WebLogic Web services. For more information, see "Associate a WS-Policy file with a Web Service" in the *WebLogic Server Administration Console Online Help*.

Attaching Oracle WSM Policies to WebLogic Web Service Clients

You attach policies to WebLogic Web service clients at design time, using JAX-WS Stubs. For more information, see "Using Oracle Web Service Security Policies" in *Securing Web Services for Oracle WebLogic Server*.

Attaching WebLogic Web Service Policies to WebLogic Web Services

You attach policies to WebLogic Web services at both design time and after the Web service has been deployed.

- At design time, use the `@Policy` and `@Policies` JWS annotations in your JWS file to associate policy files with your Web service. You can associate any number of policy files with a Web service, although it is up to you to ensure that the assertions do not contradict each other. You can specify a policy file at the class level of your JWS file. For more information, see the following sections:
  - *Securing WebLogic Web Services for Oracle WebLogic Server*.
  - "Using Policies with Web Services" in "Developing with Web Services" in the Oracle JDeveloper online help.
- After the Web service has been deployed, use the Oracle WebLogic Server Administration Console to attach WebLogic Web service policies to WebLogic Web services. For more information, see "Associate a WS-Policy file with a Web Service" in the *WebLogic Server Administration Console Online Help*.

Attaching WebLogic Web Service Policies to WebLogic Web Service Clients

You attach policies to WebLogic Web service clients at design time, using JAX-WS Stubs. For more information, see "Using a Client-side Security Policy File" in *Securing Web Services for Oracle WebLogic Server*.

Part V

Reference

Part V contains the following chapters:

- Appendix A, "Web Service Security Standards"
- Appendix B, "Predefined Policies"
- Appendix C, "Predefined Assertion Templates"
- Appendix D, "Schema Reference for Predefined Assertions"
- Appendix E, "Schema Reference for Custom Assertions"

A Web Service Security Standards

Note: This appendix summarizes the security standards for SOA, ADF, and WebCenter services. For a description of standards for WebLogic Web services, see "Standards Supported by WebLogic Web Services" in *Oracle Fusion Middleware Introducing WebLogic Web Services for Oracle WebLogic Server*

Security standards are implemented in non-XML frameworks at the transport level, and in XML frameworks at the application level.

The following sections describe the standards that are key to providing secure and manageable SOA environments at both the transport and application levels.

- Web Services Interoperability Organization—Basic Security Profile
- Transport Layer Security—SSL
- XML Encryption (Confidentiality)
- XML Signature (Integrity, Authenticity)
- WS-Security
- WS-Security Tokens
- WS-Policy
- WS-SecurityPolicy
- Web Services Addressing (WS-Addressing)
- WS-ReliableMessaging

See Also: For a complete list of standards supported by Oracle WebLogic Web services, see "Standards Supported by WebLogic Web Services" in Introducing WebLogic Web Services for Oracle WebLogic Server.

Web Services Interoperability Organization—Basic Security Profile

Oracle considers interoperability of Web services platforms to be more important than providing support for all possible edge cases of the Web services specifications. Oracle complies with the following specification from the Web Services Interoperability Organization and considers it to be the baseline for Web services interoperability:

- *Basic Security Profile 1.0*:
  http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html

Transport Layer Security—SSL

Secure Sockets Layer (SSL), also known as Transport Layer Security (TLS), is the most widely used transport-layer data-communication protocol. SSL provides the following:

- Authentication—communication is established between two trusted parties.
- Message confidentiality—data exchanged is encrypted.
- Message integrity—data is checked for corruption.
- Secure key exchange between client and server

SSL can be used in three modes:

- No authentication: Neither the client nor the server authenticates itself to the other. No certificates are sent or exchanged. In this case, only confidentiality (encryption/decryption) is used.
- One-way authentication (or server authentication): Only the server authenticates itself to the client. The server sends the client a certificate verifying that the server is authentic. This is typically the approach used for Internet transactions such as online banking.
- Two-way authentication (or bilateral authentication): Both client and server authenticate themselves to each other by sending certificates to each other. This approach is necessary to prevent attacks from occurring between a proxy and a Web service endpoint.

SSL uses a combination of secret-key and public-key cryptography to secure communications. SSL traffic uses secret keys for encryption and decryption, and the exchange of public keys is used for mutual authentication of the parties involved in the communication.

XML Encryption (Confidentiality)

The XML encryption specification describes a process for encrypting data and representing the result in XML. Specifically, XML encryption defines:

- How digital content is encrypted and decrypted.
- How the encryption key information is passed to a recipient.
- How encrypted data is identified to facilitate encryption.

An XML document may be encrypted as a whole or in part.

Example A–1 illustrates credit card data represented in XML.

Example A–1 XML Representation of Credit Card Data

```
<PaymentInfo xmlns="http://www.example.com/payment">
  <CreditCard>
    <Name>John Smith</Name>
    <CreditCardNumber>4019 2445 0277 5567</NCreditCardNumber>
    <Limit>5000</Limit>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
```

```
</PaymentInfo>
```

Example A–2 illustrates the same XML snippet with the credit card number encrypted and represented by a cipher value.

Example A–2 XML Representation of Encrypted Credit Card Data

```
<PaymentInfo xmlns='http://www.example.com/payment">
  <CreditCard>
    <Name>John Smith</Name>
    <CreditcardNumber>
      <EncryptedData xmlns="http://www..." Type="http://www...">
        <CipherData>
          <CipherValue>A23B4...5C56</CipherValue>
        </CipherData>
      </EncryptedData>
    <Limit>5000</Limit>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

See Also:

For more information about XML encryption, see "XML Encryption Syntax and Processing" specification at:

http://www.w3.org/TR/xmlenc-core

XML Signature (Integrity, Authenticity)

The XML Signature specification describes signature processing rules and syntax. XML Signature binds the sender's identity (or "signing entity") to an XML document. The document is signed using the sender's private key; the signature is verified using the sender's public key.

Signing and signature verification can be done using asymmetric or symmetric keys. XML Signature also ensures non-repudiation of the signing entity, that is, it provides proof that messages have not been altered since they were signed.

A signature can apply to a whole document or just part of a document, as shown in the following example.

Example A–3 XML Representation of Signed Data

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<!-- The signedInfo element allows us to sign any portion of a
 document -->
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www..."/>
    <SignatureMethod Algorithm="http://www..."/>
    <Reference URI="#Body">
      <DigestMethod Algorithm="http://www..."/>
      <DigestValue>o+jtqlieRtF6DrUb...X8O9M/CmySg</DigestValue>
    </Reference>
  </SignedInfo>
  <!-- Following is the result of running the algorithm over the
  document. If changes are made to the document, the SignatureValue is
  changed. The security application verifies the SignatureValue,
  extracts the X.509 cert and uses it to authenticate the user -->
  <SignatureValue>oa+ttbsvSFi...EtRD2oNC5</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <!-- Following is the public key that matches the private key
      that signs the document -->
      <RSAKeyValue>
        <Modulus>5TT/oolzTiP++Ls6GLQUM8xoFFrAlZQ...</Modulus>
        <Exponent>EQ==</Exponent>
      </RSAKeyValue>
    </KeyValue>
    <!-- Following is the certificate -->
    <X509Data>
      <X509Certificate>wDCCAXqgAwIBAgI...</X509Certificate>
    </X509Data>
  </KeyInfo>
</Signature>
```

See Also:

For more information about XML Signature, see the "XML Signature Syntax and Processing" specification at:

http://www.w3.org/TR/xmldsig-core

WS-Security

Web Services Security (WS-Security) specifies SOAP security extensions that provide confidentiality using XML Encryption and data integrity using XML Signature. WS-Security also includes profiles that specify how to insert different types of binary and XML security tokens in WS-Security headers for authentication and authorization purposes. WS-Security token profiles are described in the following sections

See Also:

For more information about WS-Security and its specification, see:

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

WS-Security Tokens

Web services security supports the following security tokens:

- Username—defines how a Web service consumer can supply a username as a credential for authentication). For more information, see "Username" on page A-4
- X.509 certificate—a signed data structure designed to send a public key to a receiving party. For more information, see "X.509 Certificate" on page A-5
- Kerberos ticket—a binary authentication and session token. For more information, see "Kerberos Ticket" on page A-5
- Security Assertion Markup Language (SAML) assertion—shares security information over the Internet through XML documents. For more information, see "SAML Token" on page A-5

Username

The username token carries basic authentication information. The `username-token` element propagates username and password information to authenticate the message.

The information provided in the token and the trust relationship provide the basis for establishing the identity of the user.

See Also:

For more information about the username token profile, see:

`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf`

X.509 Certificate

An X.509 digital certificate is a signed data structure designed to send a public key to a receiving party. A certificate includes standard fields such as certificate ID, issuer's Distinguished Name (DN), validity period, owner's DN, owner's public key, and so on.

Certificates are issued by certificate authorities (CA). A CA verifies an entity's identity and grants a certificate, signing it with the CA's private key. The CA publishes its own certificate which includes its public key.

Each network entity has a list of the certificates of the CAs it trusts. Before communicating with another entity, a given entity uses this list to verify that the signature of the other entity's certificate is from a trusted CA.

See Also:

For more information about the X.509 token profile, see:

`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf`

Kerberos Ticket

Kerberos is a cross-platform authentication and single sign-on system. The Kerberos protocol provides mutual authentication between two entities relying on a shared secret (symmetric keys). Kerberos uses the following terminology:

- A Principal is an identity for a user (i.e., a user is assigned a principal), or an identity for an application offering Kerberos services.
- A Realm is a Kerberos server environment; a Kerberos realm can be a domain name such as EXAMPLE.COM (by convention expressed in uppercase).

Kerberos involves a client, a server, and a trusted party to mediate between them called the Key Distribution Center (KDC). Each Kerberos realm has at least one KDC. KDCs come in different packages based on the operating platform used (for example, on Microsoft Windows, the KDC is a domain service). The Kerberos Token profile of WS-Security allows business partners to use Kerberos tokens in service-oriented architectures.

SAML Token

The Security Assertion Markup Language (SAML) is an open framework for sharing security information over the Internet through XML documents. SAML was designed to address the following:

- Limitations of web browser cookies to a single domain: SAML provides a standard way to transfer cookies across multiple Internet domains.

- Proprietary web single sign-on (SSO): SAML provides a standard way to implement SSO within a single domain or across multiple domains. This functionality is provided by the Oracle Identity Federation product.
- Federation: SAML facilitates identity management (e.g., account linking when a single user is known to multiple web sites under different identities), also supported by Oracle Identity Federation.
- Web Services Security: SAML provides a standard security token (a SAML assertion) that can be used with standard web services security frameworks (e.g., WS-Security) – This is the use of SAML that is particularly relevant to web services security, fully supported by Oracle WSM.
- Identity propagation: SAML provides a standard way to represent a security token that can be passed across the multiple steps of a business process or transaction, from browser to portal to networks of web services, also a feature supported by Oracle WSM.

The SAML framework includes 4 parts:

- Assertions: How you define authentication and authorization information.
- Protocols: How you ask (SAML Request) and get (SAML Response) the assertions you need.
- Bindings: How SAML Protocols ride on industry-standard transport (e.g., HTTP) and messaging frameworks (e.g., SOAP).
- Profiles: How SAML Protocols and Bindings combine to support specific use cases.

In the context of WS-Security, only SAML assertions are used. The protocols and bindings are provided by the WS-Security framework. SAML is widely adopted by the industry, both for browser-based federation and federation enabled by web services flows.

SAML assertions are very popular security tokens within WS-Security because they are very expressive and can help prevent man-in-the-middle and replay attacks.

Typically, a SAML assertion makes statements about a principal (a user or an application). All SAML assertions include the following common information:

- Issuer ID and issuance timestamp
- Assertion ID
- Subject
- Name
- Optional subject confirmation (for example, a public key)
- Optional conditions (under which an assertion is valid)
- Optional advice (on how an assertion was made)

SAML assertions can include three types of statements:

- Authentication statement: issued by an authentication authority upon successful authentication of a subject. It asserts that Subject S was authenticated by Means M at Time T.
- Attribute statement: issued by an attribute authority, based on policies. It asserts that Subject S is associated with Attributes A, B, etc. with values a, b, and so on.

- Authorization decision statement (deprecated in SAML 2.0, now supported by XACML): issued by an authorization authority which decides whether to grant the request by Subject S, for Action A (e.g., read, write, etc.), to Resource R (e.g., a file, an application, a Web service), given Evidence E.

SAML assertions can be embedded (i.e., a SAML assertion can contain another SAML assertion). SAML assertions can be signed (using XML Signature) and/or encrypted (using XML Encryption).

See Also:

For more information about the SAML token profile, see:

`http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf`

WS-Policy

Together with WS-Security, WS-Policy is another key industry standard for Oracle Fusion Middleware security.

A Web service provider may define conditions (or policies) under which a service is to be provided. The WS-Policy framework enables one to specify policy information that can be processed by web services applications, such as Oracle WSM.

A policy is expressed as one or more policy assertions representing a Web service's capabilities or requirements. For example, a policy assertion may stipulate that a request to a Web service be encrypted. Likewise, a policy assertion can define the maximum message size that a Web service can accept.

WS-Policy expressions are associated with various web services components using the WS-PolicyAttachment specification. WS-Policy information can be embedded in a WSDL file, thus making it easy to expose Web service policies through a UDDI registry.

WS-SecurityPolicy

WS-SecurityPolicy is part of the Web Services Secure Exchange (WS-SX) set of specifications hosted by OASIS (in addition to WS-SecurityPolicy, the WS-SX technical committee defines two other sets of specifications: WS-Trust and WS-SecureConversation, described later in this chapter).

WS-SecurityPolicy defines a set of security policy assertions used in the context of the WS-Policy framework. WS-SecurityPolicy assertions describe how messages are secured on a communication path. Oracle has contributed to the OASIS WS-SX technical committee several practical security scenarios (a subset of which is provided by Oracle WSM 11*g*). Each security scenario describes WS-SecurityPolicy policy expressions.

WS-SecurityPolicy *scenarios* describe examples of how to set up WS-SecurityPolicy policies for several security token types described in the WS-Security specification (supporting both WS-Security 1.0 and 1.1). The subset of the WS-SecurityPolicy scenarios supported by Oracle WSM 11*g* represents the most common customer use cases. Each scenario has been tested in multiple-vendor WS-Security environments.

To illustrate WS-SecurityPolicy, let's use a scenario supported by Oracle WSM: UsernameToken with plain text password. As mentioned earlier, Username token is one of the security tokens specified by WS-Security. This specific scenario uses a policy that says that a requester must send a password in a Username token to a recipient who has authority to validate that token. The password is a default requirement for the WS-Security Username Token Profile 1.1.

This scenario is only recommended when confidentiality of the password is not an issue, such as a pre-production test scenario with dummy passwords.

Example A–4 Example of WS-SecurityPolicy

```
<wsp:Policy>
  <sp:SupportingTokens>
    <wsp:Policy>
      <sp:UsernameToken/>
    </wsp:Policy>
  </sp:SupportingTokens>
</wsp:Policy>
```

An example of a message that conforms to the above stated policy is shown below.

Example A–5 Example of Message Conforming to WS-SecurityPolicy

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="...">
  <soap:Header>
    <wsse:Security soap:mustUnderstand="1" xmlns:wsse="...">
      <wsse:UsernameToken>
        <wsse:Username>Marc</wsse:Username>
        <wsse:Password Type="http://docs.oasis open.org...>
           XYZ
        </wsse:Password>
        <wsse:Nonce EncodingType="...#Base64Binary">qB...</wsse:Nonce>
        <wsu:Created>2008-01-02T00:01:03Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <Oracle xmlns=http://xmlsoap.org/Oracle>
      <text>EchoString</text>
    </Oracle>
  </soap:Body>
</soap:Envelope>
```

The example above contains a <Nonce> element and a <Created> timestamp, which, while optional, are recommended to improve security of requests against replay and other attacks. A nonce is a randomly generated (unique) number. The timestamp can be used to define the amount of time the security token is valid.

Web Services Addressing (WS-Addressing)

SOAP does not provide a standard way to specify where a message is going or how responses or faults are returned. WS-Addressing provides an XML framework for identifying web services endpoints and for securing end-to-end endpoint identification in messages.

A Web service endpoint is a resource (such as an application or a processor) to which web services messages are sent.

The following is an example using WS-Addressing (`wsa` is the namespace for WSAddressing):

Example A–6   Example of WS-Addressing

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
   <S:Header>
      <wsa:MessageID>http://example.com/xyz-abcd-123</wsa:MessageID>
      <wsa:ReplyTo>
         <wsa:Address>http://example.myClient1</wsa:Address>
      </wsa:ReplyTo>
```

WS-Addressing is transport-independent; that is, the request may be over JMS and the response over HTTP. WS-Addressing is used with other WS-* specifications, such as WS-Policy.

WS-ReliableMessaging

WS-ReliableMessaging (WS-RM) defines a framework for identifying and managing the reliable delivery of messages between Web services endpoints. WS-RM is predicated on the SOAP messaging structure (SOAP binding) and relies on WS-Security, WS-Policy, and WS-Addressing to provide reliable messaging.

WS-RM defines a reliable messaging (RM) source (the party that sends the message) and an RM destination (the party that receives the message). WS-RM mandates prerequisites, for example, trust between endpoints must be established, and the message and endpoints must be formally identified (this is achieved through the use of the complementary WS-* specifications mentioned earlier).

WS-RM Policy defines a policy assertion that leverages the WS-Policy framework in order to enable an RM destination and an RM source to describe their requirements for a given sequence.

B Predefined Policies

This appendix summarizes the predefined policies and contains the following sections:

- Security Policies
- WS-Addressing Policies
- MTOM Attachment Policies
- Reliable Messaging Policies
- Management Policies

Oracle has been instrumental in contributing to emerging standards, in particular the specifications hosted by the OASIS Web Services Secure Exchange technical committee. Oracle has contributed to the OASIS WS-SX technical committee several practical security scenarios, a subset of which are implemented in the predefined policies.

Note: For information about WebLogic Web service policies, see *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server*.

Security Policies

The following sections describe the security policies.

- Authentication Only Policies
- Message Protection Only Policies
- Message Protection and Authentication Policies
- Authorization Only Policies

Authentication Only Policies

Table B–1 summarizes the security policies that enforce authentication only, and indicates whether the token is inserted at the transport layer or SOAP header.

Table B–1 Authentication Only Policies

Client Policy	Service Policy	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss_http_token_client_policy	oracle/wss_http_token_service_policy	Yes	No	No	No
oracle/wss_username_token_client_policy	oracle/wss_username_token_service_policy	No	Yes	No	No
oracle/wss10_saml_token_client_policy	oracle/wss10_saml_token_service_policy	No	Yes	No	No
oracle/wss11_kerberos_token_client_policy	oracle/wss11_kerberos_token_service_policy	No	Yes	No	No

oracle/wss_http_token_client_policy

The wss_http_token_client_policy includes credentials in the HTTP header for outbound client requests. This policy can be enforced on any HTTP-based client.

Note: Currently only HTTP basic authentication is supported.

This policy contains the following policy assertion: oracle/wss_http_token_client_template. See "oracle/wss_http_token_client_template" on page C-2 for more information about the assertion.

For more information about configuring the policy, see "oracle/wss_http_token_client_policy" on page 10-5.

oracle/wss_http_token_service_policy

The wss_http_token_service_policy uses the credentials in the HTTP header to authenticate users against the Oracle Platform Security Services identity store. This policy can be enforced on any HTTP-based endpoint.

Note: Currently only HTTP basic authentication is supported.

This policy contains the following policy assertion: oracle/wss_http_token_service_template. See "oracle/wss_http_token_service_template" on page C-4 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_http_token_service_policy" on page 10-6.

oracle/wss_username_token_client_policy

This policy includes credentials in the WS-Security UsernameToken SOAP header for all outbound SOAP request messages. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

Note: Digest passwords are not supported in this release.

This policy contains the following policy assertion: oracle/wss_username_token_client_template. See "oracle/wss_username_token_client_template" on page C-5 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_username_token_client_policy" on page 10-6.

oracle/wss_username_token_service_policy

This policy uses the credentials in the WS-Security UsernameToken SOAP header to authenticate users. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based endpoint.

Note: Digest passwords are not supported in this release.

This policy contains the following policy assertion: oracle/wss_username_token_service_template. See "oracle/wss_username_token_service_template" on page C-7 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_username_token_service_policy" on page 10-7.

oracle/wss10_saml_token_client_policy

This policy includes SAML tokens in outbound SOAP request messages. The policy can be enforced on any SOAP-based client.

This policy contains the following policy assertion: oracle/wss10_saml_token_client_template. See "oracle/wss10_saml_token_client_template" on page C-8 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_client_policy" on page 10-8.

oracle/wss10_saml_token_service_policy

This policy authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header. The credentials in the SAML token are authenticated against a SAML login module. This policy can be enforced on any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/wss10_saml_token_service_template. See "oracle/wss10_saml_token_service_template" on page C-9 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_service_policy" on page 10-8.

oracle/wss11_kerberos_token_client_policy

This policy includes a Kerberos token in the WS-Security header in accordance with the WS-Security Kerberos Token Profile v1.1 standard. This policy is compatible with MIT and Active Directory KDCs. This policy can be enforced on any SOAP-based client.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_client_template. See "oracle/wss11_kerberos_token_with_message_protection_client_template" on page C-37 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_kerberos_token_client_policy" on page 10-9.

oracle/wss11_kerberos_token_service_policy

This policy is enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard. This policy extracts the Kerberos token from the SOAP header and authenticates the user. The container must have the Kerberos infrastructure configured through Oracle Platform Security Services. This policy is compatible with MIT and Active Directory KDCs. This policy can be attached to any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_service_template. See "oracle/wss11_kerberos_token_with_message_protection_service_template" on page C-39 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_service_policy" on page 10-8.

Message Protection Only Policies

Table B–2 summarizes the policies that enforce message protection only, and indicates whether the policy is enforced at the transport layer or SOAP header.

Table B–2 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

oracle/wss10_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses the WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_message_protection_client_template. See "oracle/wss11_message_protection_service_template" on page C-15 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_message_protection_client_policy" on page 10-10.

oracle/wss10_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

The messages are protected using WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_message_protection_service_template. See "oracle/wss10_message_protection_service_template" on page C-13 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_message_protection_service_policy" on page 10-12.

oracle/wss11_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss11_message_protection_client_template. See "oracle/wss11_message_protection_client_template" on page C-14 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_message_protection_client_policy" on page 10-13.

oracle/wss11_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss11_message_protection_service_template. See "oracle/wss11_message_protection_service_template" on page C-15 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_message_protection_service_policy" on page 10-15.

Message Protection and Authentication Policies

Table B–3 summarizes the policies that enforce both message protection and authentication but do not conform to the WS-Security 1.0 or 1.1 standard. The table indicates whether the policy is enforced at the transport layer or SOAP header.

Table B–3 Message Protection and Authentication Policies

Client Policy	Service Policy	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss_http_token_over_ssl_client_policy	oracle/wss_http_token_over_ssl_service_policy	Yes	No	Yes	No
Attach one of the following: ■ oracle/wss_saml_token_over_ssl_client_policy ■ oracle/wss_username_token_over_ssl_client_policy	oracle/wss_saml_or_username_token_over_ssl_service_policy	No	Yes	Yes	No
oracle/wss_saml_token_bearer_over_ssl_client_policy	oracle/wss_saml_token_bearer_over_ssl_service_policy	No	Yes	Yes	No
oracle/wss_saml_token_over_ssl_client_policy	oracle/wss_saml_token_over_ssl_service_policy	No	Yes	Yes	No
oracle/wss_username_token_over_ssl_client_policy	oracle/wss_username_token_over_ssl_service_policy	No	Yes	Yes	No
oracle/wss10_saml_hok_with_message_protection_client_policy	oracle/wss10_saml_hok_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss10_saml_token_with_message_integrity_client_policy	oracle/wss10_saml_token_with_message_integrity_service_policy	No	Yes	No	Yes
oracle/wss10_saml_token_with_message_protection_client_policy	oracle/wss10_saml_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy	oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy	No	Yes	No	Yes
oracle/wss10_username_id_propagation_with_msg_protection_client_policy	oracle/wss10_username_id_propagation_with_msg_protection_service_policy	No	Yes	No	Yes
oracle/wss10_username_token_with_message_protection_client_policy	oracle/wss10_username_token_with_message_protection_service_policy	No	Yes	No	Yes

Table B–3 (Cont.) Message Protection and Authentication Policies

Client Policy	Service Policy	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss10_username_token_with_message_protection_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
Attach one of the following: ▪ 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_username_token_with_message_protection_client_policy	oracle/wss11_username_token_with_message_protection_service_policy	No	Yes	No	Yes
oracle/wss11_x509_token_with_message_protection_client_policy	oracle/wss11_x509_token_with_message_protection_service_policy	No	Yes	No	Yes

oracle/wss_http_token_over_ssl_client_policy

This policy includes credentials in the HTTP header for outbound client requests and authenticates users against the Oracle Platform Security Services identity store. This policy also verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be enforced on any HTTP-based client.

Note: Currently only HTTP basic authentication is supported.

This policy contains the following policy assertion: oracle/wss_http_token_over_ssl_client_template. See "oracle/wss_http_token_over_ssl_client_template" on page C-17 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_http_token_over_ssl_client_policy" on page 10-16.

oracle/wss_http_token_over_ssl_service_policy

This policy extracts the credentials in the HTTP header and authenticates users against the Oracle Platform Security Services identity store. This policy verifies that the transport protocol is HTTPS. Requests over a non-HTTPS transport protocol are refused. This policy can be enforced on any HTTP-based endpoint.

Note: Currently only HTTP basic authentication is supported.

This policy contains the following policy assertion: oracle/wss_http_token_over_ssl_service_template. See "oracle/wss_http_token_over_ssl_service_template" on page C-19 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_http_token_over_ssl_service_policy" on page 10-17.

oracle/wss_saml_or_username_token_over_ssl_service_policy

This policy enforces message protection (integrity and confidentiality) and one of the following authentication policies, based on whether the client uses a SAML or username token, respectively:

- SAML token within WS-Security SOAP header using the sender-vouches confirmation type.
- WS-Security UsernameToken SOAP header to authenticate users against the Oracle Platform Security Services configured identity store.

This policy contains the following assertions, as an OR group—meaning either type of policy can be enforced by a client:

- oracle/wss_saml_token_over_ssl_service_template. See "oracle/wss_saml_token_over_ssl_service_template" on page C-21 for more information about the assertion.
- oracle/wss_username_token_over_ssl_service_template. See "oracle/wss_username_token_over_ssl_service_template" on page C-23 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml_token_over_ssl_service_policy" on page 10-20 and "oracle/wss_username_token_over_ssl_service_policy" on page 10-21.

oracle/wss_saml_token_bearer_over_ssl_client_policy

This policy includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method *Bearer* is created automatically. The policy also verifies that the transport protocol provides SSL message protection. This policy can be attached to any SOAP-based client.

This policy contains the following policy assertion: oracle/wss_saml_token_bearer_over_ssl_client_template. See "oracle/wss_saml_token_bearer_over_ssl_client_template" on page C-20 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml_token_bearer_over_ssl_client_policy" on page 10-18.

oracle/wss_saml_token_bearer_over_ssl_service_policy

This policy authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header. The credentials in the SAML token are authenticated against a SAML login module. The policy verifies that the transport protocol provides SSL message protection. This policy can be enforced on any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/wss_saml_token_bearer_over_ssl_service_template. See "oracle/wss_saml_token_bearer_over_ssl_service_template" on page C-21 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml_token_bearer_over_ssl_service_policy" on page 10-18.

oracle/wss_saml_token_over_ssl_client_policy

This policy includes SAML tokens in outbound WS-Security SOAP headers using the sender-vouches confirmation type. The policy verifies that the transport protocol provides SSL message protection. This policy can be enforced on any SOAP-based client.

This policy contains the following policy assertion: oracle/wss_saml_token_over_ssl_client_template. See "oracle/wss_saml_token_over_ssl_client_template" on page C-21 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml_token_over_ssl_client_policy" on page 10-19.

oracle/wss_saml_token_over_ssl_service_policy

This policy enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type. The SAML token is mapped to a user in the configured identity store. The policy verifies that the transport protocol provides SSL message protection. This policy can be enforced on any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/wss_saml_token_over_ssl_service_template. See "oracle/wss_saml_token_over_ssl_service_template" on page C-21 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_saml_token_over_ssl_service_policy" on page 10-20.

oracle/wss_username_token_over_ssl_client_policy

This policy includes credentials in the WS-Security UsernameToken header in outbound SOAP request messages. The policy verifies that the transport protocol provides SSL message protection. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

Note: Digest passwords are not supported in this release.

This policy contains the following policy assertion: oracle/wss_username_token_over_ssl_client_template. See "oracle/wss_username_token_over_ssl_client_template" on page C-21 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_username_token_over_ssl_client_policy" on page 10-20.

oracle/wss_username_token_over_ssl_service_policy

This policy uses the credentials in the WS-Security UsernameToken SOAP header to authenticate users against the Oracle Platform Security Services configured identity store. The policy verifies that the transport protocol provides SSL message protection. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based endpoint.

Note: Digest passwords are not supported in this release.

This policy contains the following policy assertion: oracle/wss_username_token_over_ssl_service_template. See "oracle/wss_username_token_over_ssl_service_template" on page C-23 for more information about the assertion.

For information about configuring the policy, see "oracle/wss_username_token_over_ssl_service_policy" on page 10-21.

oracle/wss10_saml_hok_with_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) and SAML holder of key based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with holder of key confirmation.

The policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_saml_hok_with_message_protection_client_template. See "oracle/wss10_saml_hok_with_message_protection_service_template" on page C-27 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_hok_token_with_message_protection_client_policy" on page 10-22.

oracle/wss10_saml_hok_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and SAML holder of key based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_saml_hok_with_message_protection_service_template. See "oracle/wss10_saml_hok_with_message_protection_service_template" on page C-27 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_hok_token_with_message_protection_service_policy" on page 10-23.

oracle/wss10_saml_token_with_message_integrity_client_policy

This policy provides message-level integrity and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with sender vouches confirmation.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies and SHA-1 hashing algorithm for message integrity. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_client_template. See "oracle/wss10_saml_token_with_message_protection_client_template" on page C-28 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_with_message_integrity_client_policy" on page 10-24.

oracle/wss10_saml_token_with_message_integrity_service_policy

This policy enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. It extracts the SAML token from the WS-Security binary security token or the current Java Authentication and Authorization Service (JAAS) subject, and uses those credentials to validate users against the Oracle Platform Security Services identity store.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies and SHA-1 hashing algorithm for message integrity. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_service_template. See "oracle/wss10_saml_token_with_message_protection_service_template" on page C-30 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_with_message_integrity_service_policy" on page 10-25.

oracle/wss10_saml_token_with_message_protection_client_policy

This policy provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_client_template. See "oracle/wss10_saml_token_with_message_protection_client_template" on page C-28 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_with_message_protection_client_policy" on page 10-25.

oracle/wss10_saml_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature. It extracts the SAML token from the WS-Security binary security token, and uses those credentials to validate users against the Oracle Platform Security Services identity store.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_service_template. See "oracle/wss10_saml_token_with_message_protection_service_template" on page C-30 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_with_message_protection_service_policy" on page 10-27.

oracle/wss10_saml_token_with_message_protection_ski_basic256_client_policy

This policy provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

The policy uses WS-Security's Basic 256 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-256 bit encryption. This policy uses Subject Key Identifier (ski) reference mechanism for encryption key in the request and for both signature and encryption keys in the response. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_client_template. See "oracle/wss10_saml_token_with_message_protection_client_template" on page C-28 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_with_message_protection_client_policy" on page 10-25.

oracle/wss10_saml_token_with_message_protection_ski_basic256_service_policy

This policy enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature. It extracts the SAML token from the WS-Security binary security token, and uses those credentials to validate users against the Oracle Platform Security Services identity store.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

The policy uses WS-Security's Basic 256 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-256 bit encryption. This policy uses Subject Key Identifier (ski) reference mechanism for encryption key in the request and for both signature and encryption keys in the response. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54

This policy contains the following policy assertion: oracle/wss10_saml_token_with_message_protection_service_template. See "oracle/wss10_saml_token_with_message_protection_service_template" on page C-30 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_saml_token_with_message_protection_service_policy" on page 10-27.

oracle/wss10_username_id_propagation_with_msg_protection_client_policy

This policy provides message protection (integrity and confidentiality) and identity propagation for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Credentials (only username) are included in outbound SOAP request messages via a WS-Security UsernameToken header. No password is included.This policy can be enforced on any SOAP-based client.

Message protection is provided using WS-Security's Basic128 suite of asymmetric key technologies. Specifically RSA key mechanisms for confidentiality, SHA-1 hashing algorithm for integrity and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_client_template. See "oracle/wss10_username_token_with_message_protection_client_template" on page C-31 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_username_id_propagation_with_msg_protection_client_policy" on page 10-30.

oracle/wss10_username_id_propagation_with_msg_protection_service_policy

This policy enforces message level protection (i.e., integrity and confidentiality) and identity propagation for inbound SOAP requests using mechanisms described in WS-Security 1.0. This policy can be enforced on any SOAP-based endpoint.

Message protection is provided using WS-Security 1.0's Basic128 suite of asymmetric key technologies. Specifically RSA key mechanisms for confidentiality, SHA-1 hashing algorithm for integrity and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_username_id_propagation_with_msg_protection_service_template. See "oracle/wss10_username_token_with_message_protection_service_template" on page C-34 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_username_id_propagation_with_msg_protection_service_policy" on page 10-31.

oracle/wss10_username_token_with_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_client_template. See "oracle/wss11_username_token_with_message_protection_client_template" on page C-43 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_username_token_with_message_protection_client_policy" on page 10-32.

oracle/wss10_username_token_with_message_protection_service_policy

This policy enforces message protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based endpoint.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_service_template. See "oracle/wss11_username_token_with_message_protection_service_template" on page C-46 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_username_token_with_message_protection_service_policy" on page 10-33.

oracle/wss10_username_token_with_message_protection_ski_basic256_client_policy

This policy provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

This policy uses WS-Security's Basic 256 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-256 bit encryption. This policy uses Subject Key Identifier (ski) reference mechanism for encryption key in the request and for both signature and encryption keys in the response. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_client_template. See "oracle/wss11_username_token_with_message_protection_client_template" on page C-43 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_username_token_with_message_protection_client_policy" on page 10-32.

oracle/wss10_username_token_with_message_protection_ski_basic256_service_policy

This policy enforces message protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based endpoint.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

This policy uses WS-Security's Basic 256 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-256 bit encryption. This policy uses Subject Key Identifier (ski) reference mechanism for encryption key in the request and for both signature and encryption keys in the response. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_username_token_with_message_protection_service_template. See "oracle/wss11_username_token_with_message_protection_service_template" on page C-46 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_username_token_with_message_protection_service_policy" on page 10-33.

oracle/wss10_x509_token_with_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) and certificate credential population for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_x509_token_with_message_protection_client_template. See "oracle/wss11_x509_token_with_message_protection_client_template" on page C-46 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_x509_token_with_message_protection_client_policy" on page 10-35.

oracle/wss10_x509_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

This policy uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. For more information about the available algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss10_x509_token_with_message_protection_service_template. See "oracle/wss11_x509_token_with_message_protection_service_template" on page C-48 for more information about the assertion.

For information about configuring the policy, see "oracle/wss10_x509_token_with_message_protection_service_policy" on page 10-36.

oracle/wss11_kerberos_token_with_message_protection_client_policy

This policy includes a Kerberos token in the WS-Security header, and uses Kerberos keys to guarantee message integrity and confidentiality, in accordance with the WS-Security Kerberos Token Profile v1.1 standard. This policy is compatible with MIT KDC only. This policy can be enforced on any SOAP-based client.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_with_message_protection_client_template. See "oracle/wss11_kerberos_token_with_message_protection_client_template" on page C-37 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_kerberos_token_with_message_protection_client_policy" on page 10-37.

oracle/wss11_kerberos_token_with_message_protection_service_policy

This policy is enforced in accordance with the WS-Security Kerberos Token Profile v1.1 standard. This policy is compatible with MIT KDC only. This policy can be attached to any SOAP-based endpoint.

This policy extracts the Kerberos token from the SOAP header and authenticates the user, and it enforces message integrity and confidentiality using Kerberos keys. The container must have the Kerberos infrastructure configured through Oracle Platform Security Services.

This policy contains the following policy assertion: oracle/wss11_kerberos_token_with_message_protection_service_template. See "oracle/wss11_kerberos_token_with_message_protection_service_template" on page C-39 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_kerberos_token_with_message_protection_service_policy" on page 10-38.

oracle/wss11_saml_token_with_message_protection_client_policy

This policy enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1. A SAML token is included in the SOAP message for use in SAML based authentication with sender vouches confirmation.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss11_saml_token_with_message_protection_client_template. See "oracle/wss11_saml_token_with_message_protection_client_template" on page C-40 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_saml_token_with_message_protection_client_policy" on page 10-39.

oracle/wss11_saml_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard. It extracts the SAML token from the WS-Security binary security token, and uses those credentials to validate users against the Oracle Platform Security Services identity store.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss11_saml_token_with_message_protection_service_template. See "oracle/wss11_saml_token_with_message_protection_service_template" on page C-42 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_saml_token_with_message_protection_service_policy" on page 10-40.

oracle/wss11_saml_or_username_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and one of the following authentication policies, based on whether the client uses a SAML or username 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.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following assertions, as an OR group—meaning either type of policy can be enforced by a client:

- oracle/wss11_saml_token_with_message_protection_service_template. See "oracle/wss11_saml_token_with_message_protection_service_template" on page C-42 for more information about the assertion.
- oracle/wss11_username_token_with_message_protection_service_template. See "oracle/wss11_username_token_with_message_protection_service_template" on page C-46 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_saml_token_with_message_protection_service_policy" on page 10-40 and "oracle/wss11_username_token_with_message_protection_service_policy" on page 10-42.

oracle/wss11_username_token_with_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard. Both plain text and digest mechanisms are supported. This policy can be attached to any SOAP-based client.

Note: Digest passwords are not supported in this release.

The Web service consumer inserts username and password credentials, and signs and encrypts the outgoing SOAP message. The Web service provider decrypts and verifies the message and the signature.

In order to prevent replay attacks, the assertion provides the option to include time stamps and verification by the Web service provider. The message can be protected with ciphers of different strengths.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss11_username_token_with_message_protection_client_template. See "oracle/wss11_username_token_with_message_protection_client_template" on page C-43 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_username_token_with_message_protection_client_policy" on page 10-41.

oracle/wss11_username_token_with_message_protection_service_policy

This policy enforces message protection (integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard. Both plain text and digest mechanisms are supported.

Note: Digest passwords are not supported in this release.

The Web service consumer inserts username and password credentials, and signs and encrypts the outgoing SOAP message. The Web service provider decrypts and verifies the message and the signature. This policy can be attached to any SOAP-based endpoint.

In order to prevent replay attacks, the assertion provides the option to include time stamps and verification by the Web service provider. The message can be protected with ciphers of different strengths.

Note: Digest passwords are not supported in this release.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss11_username_token_with_message_protection_service_template. See "oracle/wss11_username_token_with_message_protection_service_template" on page C-46 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_username_token_with_message_protection_service_policy" on page 10-42.

oracle/wss11_x509_token_with_message_protection_client_policy

This policy provides message protection (integrity and confidentiality) and certificate-based authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures. For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss11_x509_token_with_message_protection_client_template. See "oracle/wss11_x509_token_with_message_protection_client_template" on page C-46 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_x509_token_with_message_protection_client_policy" on page 10-43.

oracle/wss11_x509_token_with_message_protection_service_policy

This policy enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

This policy uses the symmetric key technology for signing and encryption, and the WS-Security's Basic 128 suite of asymmetric key technology for endorsing signatures.

For more information about the available asymmetric algorithms for message protection, see "Supported Algorithm Suites" on page C-54.

This policy contains the following policy assertion: oracle/wss11_x509_token_with_message_protection_service_template. See "oracle/wss11_x509_token_with_message_protection_service_template" on page C-48 for more information about the assertion.

For information about configuring the policy, see "oracle/wss11_x509_token_with_message_protection_service_policy" on page 10-43.

Authorization Only Policies

Table B–1 summarizes the security policies that enforce authorization, and indicates whether the policy is enforced at the transport layer or SOAP header.

Note: The authorization polices can follow any authentication policy where the Subject is established.

You cannot attach both a permitall and denyall policy to the same Web service.

Table B–4 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/binding_authorization_denyall_policy

This policy provides simple role-based authorization for the request based on the authenticated Subject at the SOAP binding level. This policy denies all users with any roles. It should follow an authentication policy where the Subject is established and can be attached to any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/binding_authorization_template. See "oracle/binding_authorization_template" on page C-49 for more information about the assertion.

For information about configuring the policy, see "oracle/binding_authorization_denyall_policy" on page 10-47.

oracle/binding_authorization_permitall_policy

This policy provides a simple role-based authorization for the request based on the authenticated Subject at the SOAP binding level. This policy permits all users with any roles. It should follow an authentication policy where the Subject is established and can be attached to any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/binding_authorization_template. See "oracle/binding_authorization_template" on page C-49 for more information about the assertion.

For information about configuring the policy, see "oracle/binding_authorization_permitall_policy" on page 10-48.

oracle/binding_permission_authorization_policy

This policy provides simple permission-based authorization for the request based on the authenticated Subject at the SOAP binding level. This policy ensures that the Subject has permission to perform the operation. This policy should follow an authentication policy where the Subject is established and can be attached to any SOAP-based endpoint.

This policy contains the following policy assertion: oracle/binding_permission_authorization_template. See "oracle/component_permission_authorization_template" on page C-52 for more information about the assertion.

For information about configuring the policy, see "oracle/binding_permission_authorization_policy" on page 10-49.

oracle/component_authorization_denyall_policy

This policy provides simple role-based authorization for the request based on the authenticated Subject at the SOAP binding level. This policy denies all users with any roles. It should follow an authentication policy where the Subject is established and can be attached to any SCA-based endpoint.

This policy contains the following policy assertion: oracle/component_authorization_template. See "oracle/component_authorization_template" on page C-51 for more information about the assertion.

For information about configuring the policy, see "oracle/component_authorization_denyall_policy" on page 10-50.

oracle/component_authorization_permitall_policy

This policy provides a simple role-based authorization policy based on the authenticated Subject. This policy permits all users with any roles. It should follow an authentication policy where the Subject is established and can be attached to any SCA-based endpoint.

This policy contains the following policy assertion: oracle/component_authorization_template. See "oracle/component_authorization_template" on page C-51 for more information about the assertion.

For information about configuring the policy, see "oracle/binding_authorization_permitall_policy" on page 10-48.

oracle/component_permission_authorization_policy

This policy provides a permission-based authorization policy based on the authenticated Subject. This policy ensures that the Subject has permission to perform the operation. This policy should follow an authentication policy where the Subject is established and can be attached to any SCA-based endpoint.

This policy contains the following policy assertion: oracle/component_permission_authorization_template. See "oracle/component_permission_authorization_template" on page C-52 for more information about the assertion.

For information about configuring the policy, see "oracle/component_permission_authorization_policy" on page 10-52.

WS-Addressing Policies

This section describes the predefined WS-Addressing policies.

Note: WS-Addressing policies are not supported for WebLogic Web services.

oracle/wsaddr_policy

This policy causes the platform to check inbound messages for the presence of WS-Addressing headers conforming to the W3C 2005 Final WS-Addressing Policy standard. In addition, it causes the platform to include a WS-Addressing header in outbound SOAP messages. For information about configuring the policy, see "oracle/wsaddr_policy" on page 10-53.

MTOM Attachment Policies

This section describes the predefined MTOM policies.

Note: MTOM policies are not supported for WebLogic Web services.

oracle/wsmtom_policy

This Message Transmission Optimization Mechanism (MTOM) policy rejects inbound messages that are not in MTOM format and verifies that outbound messages are in MTOM format. MTOM refers to specifications `http://www.w3.org/TR/2005/REC-soap12-mtom-20050125` and `http://www.w3.org/Submission/2006/SUBM-soap11mtom10-20060405` for SOAP 1.2 and SOAP 1.1 bindings, respectively. For information about configuring the policy, see "oracle/wsmtom_policy" on page 10-53.

Reliable Messaging Policies

This section describes the predefined Reliable Messaging policies.

Note: Reliable messaging policies are not supported for WebLogic Web services.

oracle/wsrm10_policy

This policy provides support for version 1.0 of the Web Services Reliable Messaging protocol. This policy can be attached to any SOAP-based client or endpoint. Full support for this feature may require additional programming. For information about configuring the policy, see "oracle/wsrm10_policy" on page 10-55.

oracle/wsrm11_policy

This policy provides support for version 1.1 of the Web Services Reliable Messaging protocol. This policy can be attached to any SOAP-based client or endpoint. Full support for this feature may require additional programming. For information about configuring the policy, see "oracle/wsrm11_policy" on page 10-56.

Management Policies

This section describes the predefined Management policies.

Note: Management policies are not supported for WebLogic Web services.

oracle/log_policy

This policy causes the request, response, and fault messages to be sent to a message log. For information about configuring the policy, see "oracle/log_policy" on page 10-56.

This policy contains the following policy assertion: oracle/log_template. See "oracle/security_log_template" on page C-53 for more information about the assertion.

C Predefined Assertion Templates

This appendix describes the predefined assertion templates that you can use to construct your policies or copy to create new policies.

This chapter contains the following sections:

- Security Assertion Templates
- Management Assertions
- Supported Algorithm Suites
- Message Signing and Encyrption Settings for Request, Response, and Fault Messages

Security Assertion Templates

The following sections describe the security assertion templates in more detail.

- Authentication Only Assertion Templates
- Message-Protection Only Assertion Template
- Message Protection and Authentication Assertion Templates
- Authorization Assertion Templates

You can jump to a specific assertion template description (client or template) using the following links (listed alphabetically):

- oracle/binding_authorization_template
- oracle/binding_permission_authorization_template
- oracle/component_authorization_template
- oracle/component_permission_authorization_template
- oracle/security_log_template
- oracle/wss_http_token_over_ssl_client_template or oracle/wss_http_token_over_ssl_service_template
- oracle/wss_http_token_client_template or oracle/wss_http_token_service_template
- oracle/wss_saml_token_bearer_over_ssl_client_template or oracle/wss_saml_token_bearer_over_ssl_service_template
- oracle/wss_saml_token_over_ssl_client_template or oracle/wss_saml_token_over_ssl_service_template

- oracle/wss_username_token_over_ssl_client_template or oracle/wss_username_token_over_ssl_service_template
- oracle/wss_username_token_client_template or oracle/wss_username_token_service_template
- oracle/wss_username_token_over_ssl_client_template or oracle/wss_username_token_over_ssl_service_template
- oracle/wss10_message_protection_client_template or oracle/wss10_message_protection_service_template
- oracle/wss10_saml_token_client_template or oracle/wss10_saml_token_service_template
- oracle/wss10_saml_token_with_message_protection_client_template or oracle/wss10_saml_token_with_message_protection_service_template
- oracle/wss10_username_token_with_message_protection_client_template or oracle/wss10_username_token_with_message_protection_service_template
- oracle/wss10_x509_token_with_message_protection_client_template or oracle/wss10_saml_token_with_message_protection_service_template
- oracle/wss11_kerberos_token_client_template or oracle/wss11_kerberos_token_service_template
- oracle/wss11_kerberos_token_with_message_protection_client_template or oracle/wss11_kerberos_token_with_message_protection_service_template
- oracle/wss11_saml_token_with_message_protection_client_template or oracle/wss11_saml_token_with_message_protection_service_template
- oracle/wss11_username_token_with_message_protection_client_template or oracle/wss11_username_token_with_message_protection_service_template
- oracle/wss11_x509_token_with_message_protection_client_template or oracle/wss11_x509_token_with_message_protection_service_template

Authentication Only Assertion Templates

Table C–59 summarizes the assertion templates that enforce authentication only, and indicates whether the token is inserted at the transport layer or SOAP header.

Table C–1 Authentication Only Assertions

Client Template	Service Template	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss_http_token_client_template	oracle/wss_http_token_service_template	Yes	No	No	No
oracle/wss_username_token_client_template	oracle/wss_username_token_service_template	No	Yes	No	No
oracle/wss10_saml_token_client_template	oracle/wss10_saml_token_service_template	No	Yes	No	No

oracle/wss_http_token_client_template

The wss_http_token_client_template assertion template includes username and password credentials in the HTTP header. You can control whether one-way or two-way authentication is required.

Settings

Table C–2 lists the settings for the wss_http_token_client_template assertion template.

Table C–2 wss_http_token_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	Authentication mechanism. Valid values include: ■ basic—Client authenticates itself by transmitting the username and password. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. ■ cert—**Not supported in this release**. Client authenticates itself by transmitting a certificate. ■ custom—**Not supported in this release**. Custom authentication mechanism.	basic
Authentication Header—Header Name	Name of the authentication header.	None
Transport Security—Require Mutual Authentication	Not applicable.	Disabled

Configurations

Table C–3 lists the identity store configurations for the wss_http_token_client_template assertion template.

Table C–3 wss_http_token_client_template Configurations

Name	Description
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to basic.credentials. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss_http_token_service_template

The wss_http_token_service_template assertion template uses the credentials in the HTTP header to authenticate users against the Oracle Platform Security Services identity store. You can control whether one-way or two-way authentication is required.

Settings

The settings for the wss_http_token_service_template are identical to those for the client version of the assertion. See Table C–2 for information on the settings.

Configurations

Table C–4 lists the identity store configurations for the wss_http_token_service_template assertion template.

Table C–4 wss_http_token_service_template Configurations

Name	Description
realm	HTTP Realm. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to owsm. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss_username_token_client_template

The wss_username_token_client_template assertion template includes authentication with username and password credentials in the WS-Security UsernameToken header. The assertion supports three types of password credentials: plain text, digest, and no password.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token.

Settings

Table C–5 lists the settings for the wss_username_token_client_template assertion template.

Table C–5 wss_username_token_client_template Settings

Name	Description	Default Value
Password Type	Type of password required. Valid values are: ■ none—No password. ■ plaintext—Unencrypted password in clear text. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. **Note**: The plaintext type is not recommended when the token propagation occurs on an unsecure channel. However, if SSL is being used as the transport channel to secure a point-to-point connection between client and server, the plaintext type can be used as the channel takes care of protecting the password.	plaintext
Nonce Required	Flag that specifies whether a nonce must be included with the username to prevent replay attacks. **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.	False
Creation Time Required	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.	False

Configurations

Table C–6 lists the identify store configurations for the wss_username_token_client_template assertion template.

Table C–6 wss_username_token_client_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to basic.credentials. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss_username_token_service_template

The wss_username_token_service_template assertion template enforces authentication with username and password credentials in the WS-Security UsernameToken SOAP header. The assertion supports three types of password credentials: plain text, digest, and no password.

Note: Digest passwords are not supported in this release.

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. See Table C–5 for information on the settings.

Configurations

Table C–7 lists the identify store configurations for the wss_username_token_service_template assertion template.

Table C–7 wss_username_token_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ▪ Value—Current value. ▪ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ▪ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ▪ Description—Description of the property.

oracle/wss10_saml_token_client_template

The wss10_saml_token_client_template assertion template includes SAML tokens in outbound SOAP request messages. The SAML token is created automatically.

Settings

Table C–8 lists the settings for the wss10_saml_token_client_template assertion template.

Table C–8 wss10_saml_token_client_template Settings

Name	Description	Default Value
Version	SAML version. The only valid value is 1.1.	1.1
Confirmation Type	Confirmation type. The only valid value is: ▪ sender-vouches—Uses the Sender Vouches SAML token for authentication.	sender-vouches

Configurations

Table C–9 lists the identity store configurations for the wss10_saml_token_client_template assertion template.

Table C–9 wss10_saml_token_client_template Configurations

Name	Description
user.roles.include	SOAP roles to be included. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to false. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to optional. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
saml.issuer.name	Name of the issuer of the SAML token. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to www.oracle.com. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to optional. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss10_saml_token_service_template

The wss10_saml_token_service_template assertion template authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header.

Settings

The settings for the wss10_saml_token_service_template are identical to the client version of the assertion. See Table C–8 for information on the settings.

Configurations

Table C–10 lists the identity store configurations for the wss10_saml_token_service_template assertion template.

Table C–10 wss10_saml_token_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ▪ Value—Current value. ▪ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ▪ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ▪ Description—Description of the property.

oracle/wss11_kerberos_token_client_template

The wss11_kerberos_token_client_template assertion template includes a Kerberos token in the WS-Security header in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

Settings

Table C–11 lists the settings for the wss11_kerberos_token_client_template assertion template.

Table C–11 wss11_kerberos_token_client_template Settings

Name	Description	Default Value
Kerberos Token Type	Type of Kerberos token. The only valid value is: gss-apreq-v5 (Kerberos Version 5 GSS-API).	gss-apreq-v5

Configurations

Table C–12 lists the identity store configurations for the wss11_kerberos_token_client_template assertion template.

Table C–12 wss11_kerberos_token_client_template Configurations

Name	Description
service.principal.name	Kerberos principal name that identifies the service. Specify the following properties: ▪ Value—Current value. ▪ Default—Default value. This value is used if Value field is not set. Defaults to HOST/localhost@EXAMPLE.COM. ▪ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ▪ Description—Description of the property.

oracle/wss11_kerberos_token_service_template

The wss11_kerberos_token_service_template assertion template enforces in accordance with the WS-Security Kerberos Token Profile v1.1 standard. It extracts the Kerberos token from the SOAP header and authenticates the user. The container must have the Kerberos infrastructure configured through Oracle Platform Security Services.

Settings

The settings for the wss11_keberos_token_service_template are identical to the client version of the assertion. See Table C–11 for information on the settings.

Configurations

Table C–13 lists the identity store configurations for the wss11_kerberos_token_service_template assertion template.

Table C–13 wss11_kerberos_token_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ▪ Value—Current value. ▪ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ▪ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ▪ Description—Description of the property.

Message-Protection Only Assertion Template

Table C–14 summarizes the assertion templates that enforce message protection only, and indicates whether the token is inserted at the transport layer or SOAP header.

Table C–14 Authentication Only Assertions

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

oracle/wss10_message_protection_client_template

The wss10_message_protection_client_template assertion template provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

Settings

Table C–15 lists the settings for the wss10_message_protection_client_template assertion template.

Table C–15 wss10_message_protection_client_template Settings

Name	Description	Default Value
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Sign Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Encryption Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-54.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Request Message Settings	See Table C–62.	N/A
Response Message Settings	See Table C–62.	N/A
Fault Message Settings	See Table C–62.	N/A

Configurations

Table C–16 lists the identity store configurations for the wss10_message_protection_client_template assertion template.

Table C–16 wss10_message_protection_client_template Configurations

Name	Description
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security runtime uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to orakey. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss10_message_protection_service_template

The wss10_message_protection_service_template assertion template provides message protection (integrity and confidentiality) for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

Settings

The settings for the wss10_message_protection_service_template are identical to the client version of the assertion. See Table C–15 for information on the settings.

Configurations

Table C–17 lists the identity store configurations for the wss10_message_protection_client_template assertion template.

Table C–17 wss10_message_protection_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss11_message_protection_client_template

The wss11_message_protection_client_template assertion template provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

Settings

Table C–18 lists the settings for the wss11_message_protection_client_template assertion template.

Table C–18 wss11_message_protection_client_template Settings

Name	Description	Default Value
Confirm Signature	Flag that specifies whether to send a signature confirmation back to the client.	True
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ■ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead.	thumbprint
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-54.	Basic128

Table C–18 (Cont.) wss11_message_protection_client_template Settings

Name	Description	Default Value
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Request Message Settings	See Table C–62.	N/A
Response Message Settings	See Table C–62.	N/A
Fault Message Settings	See Table C–62.	N/A

Configurations

Table C–19 lists the identity store configurations for the wss11_message_protection_client_template assertion template.

Table C–19 wss11_message_protection_client_template Configurations

Name	Description
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security runtime uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to orakey. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss11_message_protection_service_template

The wss11_message_protection_service_template assertion template enforces message protection (integrity and confidentiality) for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

Settings

The settings for the wss11_message_protection_service_template are identical to the client version of the assertion. See Table C–18 for information on the settings.

Configurations

Table C–20 lists the identity store configurations for the wss11_message_protection_service_template assertion template.

Table C–20 wss11_message_protection_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ▪ Value—Current value. ▪ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ▪ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ▪ Description—Description of the property.

Message Protection and Authentication Assertion Templates

Table C–21 summarizes the assertion templates that enforce both message protection and authentication, and indicates whether the token is inserted at the transport layer or SOAP header.

Table C–21 Message Protection and Authentication Assertions

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_saml_token_over_ssl_client_template	oracle/wss_saml_token_over_ssl_service_template	No	Yes	Yes	No
oracle/wss_username_token_over_ssl_client_template	oracle/wss_username_token_over_ssl_service_template	No	Yes	Yes	No
oracle/wss10_saml_hok_with_message_protection_client_template	oracle/wss10_saml_hok_with_message_protection_service_template	No	Yes	No	Yes

Table C–21 (Cont.) Message Protection and Authentication Assertions

Client Template	Service Template	Authentication Transport	Authentication SOAP	Message Protection Transport	Message Protection SOAP
oracle/wss10_saml_token_with_message_protection_client_template	oracle/wss10_saml_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_username_token_with_message_protection_client_template	oracle/wss11_username_token_with_message_protection_service_template	No	Yes	No	Yes
oracle/wss11_x509_token_with_message_protection_client_template	oracle/wss11_x509_token_with_message_protection_service_template	No	Yes	No	Yes

oracle/wss_http_token_over_ssl_client_template

The wss_http_token_over_ssl_client_template assertion template includes credentials in the HTTP header for outbound client requests and authenticates users against the Oracle Platform Security Services identity store.

Settings

Table C–22 lists the settings for the wss_http_token_over_ssl_client_template assertion template.

Table C–22 wss_http_token_over_ssl_client_template Settings

Name	Description	Default Value
Authentication Header—Mechanism	Authentication mechanism. Valid values include: ▪ basic—Client authenticates itself by transmitting the username and password. ▪ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. ▪ cert—**Not supported in this release**. Client authenticates itself by transmitting a certificate. ▪ custom—**Not supported in this release**. Custom authentication mechanism.	basic
Authentication Header—Header Name	Name of the authentication header.	None
Transport Security—Require Mutual Authentication	Flag that specifies whether two-way authentication is required. Valid values include: ▪ Enabled—The service must authenticate itself to the client, and the client must authenticate itself to the service. ▪ Disabled—One-way authentication is required. The service must authenticate itself to the client, but the client is not required to authenticate itself to the service.	Disabled

Configurations

Table C–23 lists the identity store configurations for the wss_http_token_over_ssl_client_template assertion template.

Table C–23 wss_http_token_over_ssl_client_template Configurations

Name	Description
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Specify the following properties: ▪ Value—Current value. ▪ Default—Default value. This value is used if Value field is not set. Defaults to basic.credentials. ▪ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ▪ Description—Description of the property.
role	SOAP role. Specify the following properties: ▪ Value—Current value. ▪ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ▪ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ▪ Description—Description of the property.

oracle/wss_http_token_over_ssl_service_template

The wss_http_token_over_ssl_service_template assertion template extracts the credentials in the HTTP header and authenticates users against the Oracle Platform Security Services identity store.

Settings

The settings for the wss_http_token_over_ssl_service_template assertion template are identical to the client version of the assertion. See Table C–22 for information on the settings.

Configurations

Table C–24 lists the identity store configurations for the wss_http_token_service_ template assertion template.

Table C–24 wss_http_token_over_ssl_service_template Configurations

Name	Description
realm	HTTP Realm. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to owsm. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss_saml_token_bearer_over_ssl_client_template

The wss_saml_token_bearer_over_ssl_client template assertion template includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method [*Bearer*] is created automatically.

Settings

Table C–25 lists the settings for the wss_saml_token_bearer_over_ssl_client_template assertion template.

Table C–25 wss_saml_token_bearer_over_ssl_client_template Settings

Name	Description	Default Value
Algorithm Suite	Algorithm suite used for message protection. Valid algorithm suites include: Basic128, Basic256, and TripleDES. See "Supported Algorithm Suites" on page C-54.	Basic256

Configurations

None defined.

oracle/wss_saml_token_bearer_over_ssl_service_template

The wss_saml_token_bearer_over_ssl_service_template assertion template authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.

Settings

The settings for the wss_saml_token_bearer_over_ssl_service_template assertion template are identical to the client version of the assertion. See Table C–25 for information on the settings.

Configurations

None defined.

oracle/wss_saml_token_over_ssl_client_template

The wss_saml_token_over_ssl_client_template assertion template enables the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type.

Settings

Table C–26 lists the settings for the wss_saml_token_over_ssl_client_template assertion template.

Table C–26 wss_saml_token_over_ssl_client_template Settings

Name	Description	Default Value
Algorithm Suite	Algorithm suite used for message protection. Valid algorithm suites include: Basic128, Basic256, and TripleDES. See "Supported Algorithm Suites" on page C-54.	Basic256

Configurations

None defined.

oracle/wss_saml_token_over_ssl_service_template

The wss_saml_token_over_ssl_service_template enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type.

Settings

The settings for the wss_saml_token_over_ssl_service_template assertion template are identical to the client version of the assertion. See Table C–26 for information on the settings.

Configurations

None defined.

oracle/wss_username_token_over_ssl_client_template

The wss_username_token_over_ssl_client_template assertion template includes credentials in the WS-Security UsernameToken header in outbound SOAP request messages. The assertion supports three types of password credentials: plain text, digest, and no password.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token.

Settings

Table C–27 lists the settings for the wss_username_token_over_ssl_client_template assertion template.

Table C–27 wss_username_token_over_ssl_client_template Settings

Name	Description	Default Value
Password Type	Type of password required. Valid values are: ■ none—No password. ■ plaintext—Unencrypted password in clear text. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest. **Note**: The plaintext type is not recommended when the token propagation occurs on an unsecure channel. However, if SSL is being used as the transport channel to secure a point-to-point connection between client and server, the plaintext type can be used as the channel takes care of protecting the password.	plaintext
Nonce Required	Flag that specifies whether a nonce must be included with the username to prevent replay attacks. **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.	False
Creation Time Required	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.	False
Mutual Authentication Required	Flag that specifies whether two-way authentication is required. Valid values include: ■ Enabled—Two-way authentication. The service must authenticate itself to the client, and the client must authenticate itself to the service. ■ Disabled—One-way authentication. The service must authenticate itself to the client, but the client is not required to authenticate itself to the service.	Disabled

Configurations

Table C–28 lists the identity store configurations for the wss_username_token_over_ssl_client_template assertion template.

Table C–28 wss_username_token_over_ssl_client_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services (OPSS) identity store. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to basic.credentials. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss_username_token_over_ssl_service_template

The wss_username_token_over_ssl_service_template assertion template uses the credentials in the UsernameToken WS-Security SOAP header to authenticate users against the Oracle Platform Security Services configured identity store. The assertion supports three types of password credentials: plain text, digest, and no password.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token.

Settings

The settings for the wss_username_token_over_ssl_service_template assertion template are identical to the client version of the assertion. See Table C–28 for information on the settings.

Configurations

Table C–29 lists the identity store configurations for the wss_username_token_over_ssl_service_template assertion template.

Table C–29 wss_username_token_over_ssl_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ▪ Value—Current value. ▪ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ▪ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ▪ Description—Description of the property.

oracle/wss10_saml_hok_with_message_protection_client_template

The wss10_saml_hok_with_message_protection_client_template assertion template provides message protection (integrity and confidentiality) and SAML holder of key based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

Settings

Table C–30 lists the settings for the wss10_saml_hok_with_message_protection_client_template assertion template.

Table C–30 wss10_saml_hok_with_message_protection_client_template Settings

Name	Description	Default Value
Version	SAML version. The only valid value is: 1.1.	1.1
Confirmation Type	Confirmation type. The only valid value is: holder-of-key.	holder-of-key
Is Signed	Flag that specifies whether the username is signed. The only valid value for SAML policies is: True.	True
Is Encrypted	Flag that specifies whether the username is encrypted.	False

Table C–30 (Cont.) wss10_saml_hok_with_message_protection_client_template Settings

Name	Description	Default Value
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	ski
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct
Recipient Sign Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Encryption Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-54.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Request Message Settings	See Table C–62.	N/A
Response Message Settings	See Table C–62.	N/A
Fault Message Settings	See Table C–62.	N/A

Configurations

Table C–31 lists the identity store configurations for the wss10_saml_hok_with_message_protection_client_template assertion template.

Table C–31 wss10_saml_hok_with_message_protection_client_template Configurations

Name	Description
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security runtime uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to orakey. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
saml.issuer.name	Name identifier for the issuer of the SAML token. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to www.oracle.com. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to optional. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
user.roles.include	Flag that specifies whether to include SOAP roles. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to false. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to optional. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

Table C–31 (Cont.) wss10_saml_hok_with_message_protection_client_template Configurations

Name	Description
saml.assertion.filename	Name of the of the SAML token file. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to temp. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to optional. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss10_saml_hok_with_message_protection_service_template

The wss10_saml_hok_with_message_protection_client_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_with_message_protection_service_template are identical to those for client version of the assertion. See Table C–30 for information on the settings.

Configurations

Table C–32 lists the identity store configurations for the wss10_saml_hok_with_message_protection_service_template assertion template.

Table C–32 wss10_saml_hok_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss10_saml_token_with_message_protection_client_template

The wss10_saml_token_with_message_protection_client_template assertion template provides message-level protection and SAML-based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

The Web service consumer includes a SAML token in the SOAP header, and the confirmation type is sender-vouches. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

Settings

Table C–33 lists the settings for the wss10_saml_token_with_message_protection_client_template assertion template.

Table C–33 wss10_saml_token_with_message_protection_client_template Settings

Name	Description	Default Value
Version	SAML version. The only valid value is: 1.1.	1.1
Confirmation Type	Confirmation type. The only valid value is: sender-vouches.	sender-vouches
Is Signed	Flag that specifies whether the username is signed. The only valid value for SAML policies is: True.	True
Is Encrypted	Flag that specifies whether the username is encrypted.	False
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ▪ direct—X.509 Token is included in the request. ▪ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ▪ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Sign Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Encryption Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-54.	Basic128

Table C–33 (Cont.) wss10_saml_token_with_message_protection_client_template Settings

Name	Description	Default Value
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Request Message Settings	See Table C–62.	N/A
Response Message Settings	See Table C–62.	N/A
Fault Message Settings	See Table C–62.	N/A

Configurations

Table C–34 lists the identity store configurations for the wss10_saml_token_with_message_protection_client_template assertion template.

Table C–34 wss10_saml_token_with_message_protection_client_template Configurations

Name	Description
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security runtime uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to orakey. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
user.roles.include	Flag that specifies whether to include SOAP roles. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to false. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to optional. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
saml.issuer.name	Name identifier for the issuer of the SAML token. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to www.oracle.com. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to optional. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss10_saml_token_with_message_protection_service_template

The wss10_saml_token_with_message_protection_service_template assertion template enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

The Web service consumer includes a SAML token in the SOAP header, and the confirmation type is sender-vouches. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

To prevent replay attacks, the assertion provides the option to include time stamps, SAML token limits, and their verification by the Web service provider.

Settings

The settings for the wss10_saml_token_with_message_protection_service_template are identical to those for client version of the assertion. See Table C–34 for information on the settings.

Configurations

Table C–35 lists the identity store configurations for the wss10_saml_token_with_message_protection_service_template assertion template.

Table C–35 wss10_saml_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss10_username_token_with_message_protection_client_template

The wss10_username_token_with_message_protection_client_template assertion template provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Credentials are included in the WS-Security UsernameToken header in the outbound SOAP message.

The assertion supports three types of password credentials: plain text, digest, and no password.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

Settings

Table C–36 lists the settings for the wss10_username_token_with_message_protection_client_template assertion template.

Table C–36 wss10_username_token_with_message_protection_client_template Settings

Name	Description	Default Value
Password Type	Type of password required. Valid values are: ▪ none—No password. ▪ plaintext—Unencrypted password in clear text. ▪ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest.	plaintext
Nonce Required	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.	False
Creation Time Required	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.	False
Is Signed	Flag that specifies whether the username is signed.	True
Is Encrypted	Flag that specifies whether the username is encrypted.	True
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ▪ direct—X.509 Token is included in the request. ▪ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ▪ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Sign Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Encryption Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-54.	Basic128

Table C–36 (Cont.) wss10_username_token_with_message_protection_client_template Settings

Name	Description	Default Value
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Request Message Settings	See Table C–62.	N/A
Response Message Settings	See Table C–62.	N/A
Fault Message Settings	See Table C–62.	N/A

Configurations

Table C–37 lists the identity store configurations for the wss10_username_token_with_message_protection_client_template assertion template.

Table C–37 wss10_username_token_with_message_protection_client_template Configurations

Name	Description
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Specify the following properties: ■ Value—Current value ■ Default—Default value. This value is used if the Value field is not set. Defaults to basic.credentials. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

Table C–37 (Cont.) wss10_username_token_with_message_protection_client_template Configurations

Name	Description
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security runtime uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to orakey. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss10_username_token_with_message_protection_service_template

The wss10_username_token_with_message_protection_service_template assertion template enforces message protection (integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

The assertion supports three types of password credentials: plain text, digest, and no password.

Note: Digest passwords are not supported in this release.

To protect against replay attacks, the assertion provides the option to require nonce or creation time in the username token. The SOAP message is signed and encrypted. The Web service provider decrypts the message, and verifies and authenticates the signature.

Settings

The settings for the wss10_username_token_with_message_protection_service_template assertion template are identical to the client version of the assertion. See Table C–36 for information on the settings.

Configurations

Table C–38 lists the identity store configurations for the wss10_username_token_with_message_protection_service_template assertion template.

Table C–38 wss10_username_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss10_x509_token_with_message_protection_client_template

The wss10_x509_token_with_message_protection_client template assertion template provides message protection (integrity and confidentiality) and certificate credential population for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

Settings

Table C–36 lists the settings for the wss10_x509_token_with_message_protection_client template assertion template.

Table C–39 wss10_x509_token_with_message_protection_client_template Settings

Name	Description	Default Value
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Sign Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Recipient Encryption Key Reference Mechanism	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.	direct

Table C–39 (Cont.) wss10_x509_token_with_message_protection_client_template Settings

Name	Description	Default Value
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-54.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Request Message Settings	See Table C–62.	N/A
Response Message Settings	See Table C–62.	N/A
Fault Message Settings	See Table C–62.	N/A

Configurations

Table C–40 lists the identity store configurations for the wss10_x509_token_with_message_protection_client_template assertion template.

Table C–40 wss10_x509_token_with_message_protection_client_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security runtime uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to orakey. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss10_x509_token_with_message_protection_service_template

The wss10_x509_token_with_message_protection_service_template assertion template enforces message protection (integrity and confidentiality) and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

Settings

The settings for the wss10_x509_token_with_message_protection_service_template assertion template are identical to the client version of the assertion. See Table C–39 for information on the settings.

Configurations

Table C–41 lists the identity store configurations for the wss10_x509_token_with_message_protection_service_template assertion template.

Table C–41 wss10_x509_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ▪ Value—Current value. ▪ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ▪ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ▪ Description—Description of the property.

oracle/wss11_kerberos_token_with_message_protection_client_template

The wss11_kerberos_token_with_message_protection_client_template assertion template includes a Kerberos token in the WS-Security header in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

Settings

Table C–42 lists the settings for the wss11_kerberos_token_with_message_protection_client_template assertion template.

Table C–42 wss11_kerberos_token_with_message_protection_client_template Settings

Name	Description	Default Value
Kerberos Token Type	Type of Kerberos token. The only valid value is: gss-apreq-v5 (Kerberos Version 5 GSS-API).	gss-apreq-v5
Confirm Signature	Flag that specifies whether to send a signature confirmation back to the client.	True
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ▪ direct—X.509 Token is included in the request. ▪ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ▪ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it.	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	direct
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-54.	TripleDes
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Request Message Settings	See Table C–62.	N/A
Response Message Settings	See Table C–62.	N/A
Fault Message Settings	See Table C–62.	N/A

Configurations

Table C–43 lists the identity store configurations for the wss11_kerberos_token_with_message_protection_client_template assertion template.

Table C–43 wss11_kerberos_token_with_message_protection_client_template Configurations

Name	Description
service.principal.name	Kerberos principal name that identifies the service. Specify the following properties: ▪ Value—Current value. ▪ Default—Default value. This value is used if Value field is not set. Defaults to HOST/localhost@EXAMPLE.COM. ▪ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ▪ Description—Description of the property.

oracle/wss11_kerberos_token_with_message_protection_service_template

The wss11_kerberos_token_with_message_protection_service_template assertion template enforces in accordance with the WS-Security Kerberos Token Profile v1.1 standard. It extracts the Kerberos token from the SOAP header and authenticates the user. The container must have the Kerberos infrastructure configured through Oracle Platform Security Services.

Settings

The settings for the wss11_keberos_token_with_message_protection_service_template are identical to the client version of the assertion. See Table C–42 for information on the settings.

Configurations

None required.

Table C–44 wss11_kerberos_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ▪ Value—Current value. ▪ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ▪ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ▪ Description—Description of the property.

oracle/wss11_saml_token_with_message_protection_client_template

The wss11_saml_token_with_message_protection_client_template assertion template enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests in accordance with WS-Security 1.1. A SAML token is included in the SOAP message for use in SAML based authentication with sender vouches confirmation.

Settings

Table C–45 lists the settings for the wss11_saml_token_with_message_protection_client_template assertion template.

Table C–45 wss11_saml_token_with_message_protection_client_template Settings

Name	Description	Default Value
Version	SAML version. The only valid value is: 1.1.	None
Confirmation Type	Confirmation type. Valid values include: sender-vouches.	sender-vouches.
Is Signed	Flag that specifies whether the username is signed. The only valid value for SAML policies is: True.	True
Is Encrypted	Flag that specifies whether the username is encrypted.	False
Confirm Signature	Flag that specifies whether to send a signature confirmation back to the client.	True
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ■ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead. This value is valid for Encryption Key Reference Mechanism only (described below.)	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	thumbprint
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-54.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled

Table C–45 (Cont.) wss11_saml_token_with_message_protection_client_template Settings

Name	Description	Default Value
Request Message Settings	See Table C–62.	N/A
Response Message Settings	See Table C–62.	N/A
Fault Message Settings	See Table C–62.	N/A

Configurations

Table C–45 lists the identity store configurations for the wss11_saml_token_with_message_protection_client_template assertion template.

Table C–46 wss11_saml_token_with_message_protection_client_template Configurations

Name	Description
saml.issuer.name	Name identifier for the issuer of the SAML token. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to www.oracle.com. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to optional. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security runtime uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to orakey. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss11_saml_token_with_message_protection_service_template

The wss11_saml_token_with_message_protection_service_template assertion template enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 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. See Table C–45 for information on the settings.

Configurations

Table C–44 lists the identity store configurations for the wss11_saml_token__with_ message_protection_service_template assertion template.

Table C–47 wss11_saml_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss11_username_token_with_message_protection_client_template

The ws11_username_token_with_message_protection_client_template assertion template includes authentication and message protection in accordance with the WS-Security v1.1 standard.

The Web service consumer inserts username and password credentials, and signs and encrypts the outgoing SOAP message. The Web service provider decrypts and verifies the message and the signature.

In order to prevent replay attacks, the assertion provides the option to include time stamps and verification by the Web service provider. The message can be protected with ciphers of different strengths.

Settings

Table C–48 lists the settings for the wss11_username_token_with_message_ protection_client_template assertion template.

Table C–48 wss11_username_token_with_message_protection_client_template Settings

Name	Description	Default Value
Password Type	Type of password required. Valid values are: ■ none—No password. ■ plaintext—Unencrypted password in clear text. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest.	plaintext
Nonce Required	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.	False
Creation Time Required	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.	False
Is Signed	Flag that specifies whether the username is signed.	True
Is Encrypted	Flag that specifies whether the username is encrypted.	True
Confirm Signature	Flag that specifies whether to send a signature confirmation back to the client.	True
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ■ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead.	thumbprint
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-54.	Basic256
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Request Message Settings	See Table C–62.	N/A
Response Message Settings	See Table C–62.	N/A
Fault Message Settings	See Table C–62.	N/A

Configurations

Table C–49 lists the identity store configurations for the wss11_username_token_with_message_protection_client_template assertion template.

Table C–49 wss11_username_token_with_message_protection_client_template Configurations

Name	Description
csf-key	Credential Store Key that maps to a username and password in the Oracle Platform Security Services identity store. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to basic.credentials. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security runtime uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to orakey. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss11_username_token_with_message_protection_service_template

The ws11_username_token_with_message_protection_service_template assertion template enforces authentication and message protection in accordance with the WS-Security v1.1 standard.

The Web service consumer inserts username and password credentials, and signs and encrypts the outgoing SOAP message. The Web service provider decrypts and verifies the message and the signature. In order 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. See Table C–48 for information on the settings.

Configurations

Table C–50 lists the identity store configurations for the wss11_username_token_with_message_protection_service_template assertion template.

Table C–50 wss11_username_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss11_x509_token_with_message_protection_client_template

The wss11_x509_token_with_message_protection_client_template assertion template provides message protection (integrity and confidentiality) and certificate-based authentication for outbound SOAP requests in accordance with the WS-Security 1.1 standard. Credentials are included in the WS-Security binary security token of the SOAP message.]

Settings

Table C–51 lists the settings for the wss11_x509_token_with_message_protection_client_template assertion template.

Table C–51 wss11_x509_token_with_message_protection_client_template Settings

Name	Description	Default Value
Confirm Signature	Flag that specifies whether to send a signature confirmation back to the client.	True
Sign Key Reference Mechanism	Mechanism used when signing the request. Valid values include: ■ direct—X.509 Token is included in the request. ■ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ■ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ■ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead. This value is valid for Encryption Key Reference Mechanism only (described below.)	direct
Encryption Key Reference Mechanism	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.	thumbprint
Algorithm Suite	Algorithm suite used for message protection. See "Supported Algorithm Suites" on page C-54.	Basic128
Include Timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.	Enabled
Request Message Settings	See Table C–62.	N/A
Response Message Settings	See Table C–62.	N/A
Fault Message Settings	See Table C–62.	N/A

Configurations

Table C–52 lists the identity store configurations for the wss11_x509_token_with_message_protection_client_template assertion template.

Table C–52 wss11_x509_token_with_message_protection_client_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.
keystore.recipient.alias	Keystore alias associated with the peer certificate. The security runtime uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to orakey. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to required. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

oracle/wss11_x509_token_with_message_protection_service_template

The wss11_x509_token_with_message_protection_service_template assertion template enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard. The certificate is extracted from the WS-Security binary security token header, and the credentials in the certificate are validated against the Oracle Platform Security Services identity store.

Settings

The settings for the wss11_x509_token_with_message_protection_service_template are identical to the client version of the assertion. See Table C–51 for information on the settings.

Configurations

Table C–53 lists the identity store configurations for the wss11_x509_token_with_message_protection_service_template assertion template.

Table C–53 wss11_x509_token_with_message_protection_service_template Configurations

Name	Description
role	SOAP role. Specify the following properties: ■ Value—Current value. ■ Default—Default value. This value is used if Value field is not set. Defaults to ultimateReceiver. ■ Type—Specifies one of the following values: - Constant—Property cannot be overridden. - Required—Property is required and can be overridden. - Optional—Property is optional and can be overridden. This value defaults to constant. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13. ■ Description—Description of the property.

Authorization Assertion Templates

Table C–54 summarizes assertion templates that are used for authorization. Each authorization assertion template must follow an authentication assertion template.

Table C–54 Authorization Assertion Templates

Service Template	Description
oracle/binding_authorization_template	Provides simple role-based authorization for the request based on the authenticated subject at the SOAP binding level.
oracle/binding_permission_authorization_template	Provides simple permission-based authorization for the request based on the authenticated subject at the SOAP binding level.
oracle/component_authorization_template	Provides simple role-based authorization for the request based on the authenticated subject at the SOA component level.
oracle/component_permission_authorization_template	Provides simple permission-based authorization for the request based on the authenticated subject at the SOA component level.

oracle/binding_authorization_template

The binding_authorization_template assertion template provides simple role-based authorization for the request based on the authenticated subject at the SOAP binding level. It should follow an authentication assertion template.

Settings

Table C–55 lists the settings for the binding_authorization_template assertion template.

Table C–55 binding_authorization_template Settings

Name	**Description**	**Default Value**
Action Pattern	Action or Web service operation for which authorization checks are performed. This value can be a comma-separated list of values. This field accepts wildcards. For example, `validate,amountAvailable`.	actionMatchPatte rn
Resource Pattern	Name of the resource for which authorization checks are performed. This field accepts wildcards. For example, if the namespace of the Web service is `http://project11` and the service name is `CreditValidation`, the resource name is `http://project11/CreditValidation`.	resourceMatchPat tern
Authorization Setting	Specifies the roles that are authorized. The valid values are: ■ Permit All—Permit users with any roles. ■ Deny All—Deny all users with roles. ■ Selected Roles—Permit selected roles. To add roles: 1. Click **Add**. 2. To add roles, click the checkbox next to each role you want to add in the Roles Available column and click **Move**. To add all roles, click **Move All**. To remove roles, click the checkbox next to each role you want to remove in the Roles Selected to Add column, and click **Remove**. To remove all roles, click **Remove All**. To search for roles, enter a search string in the Role Name search box and click the go arrow. The Roles Available column is updated to include only those roles that match the search string. 3. Click **OK**. To delete roles: 1. Select the role that you want to delete in the Selected Roles list. 2. Click **Delete**.	Selected Roles

Configurations

None defined.

oracle/binding_permission_authorization_template

The binding_permission_authorization_template assertion provides simple permission-based authorization for the request based on the authenticated subject at the SOAP binding level. It should follow an authentication assertion.

Note: You should be careful when using permission-based policies with EJBs as the security permissions specified in system-jazn-data.xml will be relaxed beyond a single invocation of the service operation.

Settings

Table C–56 lists the settings for the binding_permission_authorization_template assertion template.

Table C–56 binding_permission_authorization_template Settings

Name	Description	Default Value
Constraint Pattern	Reserved for future use.	N/A
Action Pattern	Action or Web service operation for which permission-based checks are performed. This value can be a comma-separated list of values. This field accepts wildcards. For example, `validate,amountAvailable`.	*
Resource Pattern	Name of the resource for which permission-based checks are performed. This field accepts wildcards. For example, if the namespace of the Web service is `http://project11` and the service name is `CreditValidation`, the resource name is `http://project11/CreditValidation`.	*
Permission Check Class	Class used for the permission-based checking. For example, `oracle.wsm.security.WSFuncPermission`.	N/A

Configurations

None defined.

oracle/component_authorization_template

The component_authorization_template assertion provides simple role-based authorization for the request based on the authenticated subject at the SOA component level. It should follow an authentication assertion.

Settings

Table C–57 lists the settings for the component_authorization_template assertion template.

Table C–57 component_authorization_template Settings

Name	Description	Default Value
Authorization Setting	Specifies the roles that are authorized. The valid values are: ■ Permit All—Permit users with any roles. ■ Deny All—Deny all users with roles. ■ Selected Roles—Permit selected roles. To add roles: 1. Click **Add**. 2. To add roles, click the checkbox next to each role you want to add in the Roles Available column and click **Move**. To add all roles, click **Move All**. To remove roles, click the checkbox next to each role you want to remove in the Roles Selected to Add column, and click **Remove**. To remove all roles, click **Remove All**. To search for roles, enter a search string in the Role Name search box and click the go arrow. The Roles Available column is updated to include only those roles that match the search string. 3. Click **OK**. To delete roles: 1. Select the role that you want to delete in the Selected Roles list. 2. Click **Delete**.	Selected Roles

Configurations

None defined.

oracle/component_permission_authorization_template

The component_permission_authorization_template assertion provides simple permission-based authorization for the request based on the authenticated subject at the SOA component level. It should follow an authentication assertion.

Note: You should be careful when using permission-based policies with EJBs as the security permissions specified in system-jazn-data.xml will be relaxed beyond a single invocation of the service operation.

Settings

Table C–58 lists the settings for the component_permission_authorization_template assertion template.

Table C–58 component_permission_authorization_template Settings

Name	Description	Default Value
Constraint Pattern	Reserved for future use.	N/A
Action Pattern	Action or Web service operation for which permission-based checks are performed. This value can be a comma-separated list of values. This field accepts wildcards. For example, `validate,amountAvailable`.	*
Resource Pattern	Name of the resource for which permission-based checks are performed. This field accepts wildcards. For example, if the composite name of the Web service is `HelloWorld` and the service name is `Hello`, the resource name is `HelloWorld/Hello`.	*
Permission Check Class	Class used for the permission-based checking. For example, `oracle.wsm.security.WSFunctionPermission`.	N/A

Configurations

None defined.

Management Assertions

Table C–59 summarizes the management assertion templates.

Table C–59 Management Assertion Templates

Name	Description
oracle/security_log_template	Provides simple role-based authorization for the request based on the authenticated subject.

oracle/security_log_template

The security_log_template assertion template provides a logging assertion template that can be attached to any binding or component.

Note: It is recommended that the logging assertion be used for debugging and auditing purposes only.

Settings

Table C–60 lists the settings for the security_log_template assertion template.

Table C–60 security_log_template Settings

Name	Description	Default Value
Request	Requirements for logging request messages. The valid values are: ■ all—Log the entire SOAP message. ■ header—Log SOAP header information only. ■ soap_body—Log SOAP body information only. ■ soap_envelope—Log SOAP envelope information only.	all
Response	Requirements for logging response messages. The valid values are the same as for Request above.	soap_body

Configurations

None defined.

Supported Algorithm Suites

Table C–61 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.

Table C–61 Supported Algorithm Suites

Algorithm Suite	Digest	Encryption	Symmetric Key Wrap	Asymmetric Key Wrap	Encrypted Key Derivation	Signature Key Derivation	Minimum Signature Key Length
Basic256	Sha1	Aes256	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256
Basic192	Sha1	Aes192	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192
Basic128	Sha1	Aes128	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128
TripleDes	Sha1	TripleDes	KwTripleDes	KwRsaOaep	PSha1L192	PSha1L192	192
Basic256Rsa15	Sha1	Aes256	KwAes256	KwRsa15	PSha1L256	PSha1L192	256
Basic192Rsa15	Sha1	Aes192	KwAes192	KwRsa15	PSha1L192	PSha1L192	192
Basic128Rsa15	Sha1	Aes128	KwAes128	KwRsa15	PSha1L128	PSha1L128	128
TripleDesRsa15	Sha1	TripleDes	KwTripleDes	KwRsa15	PSha1L192	PSha1L192	192

Message Signing and Encyrption Settings for Request, Response, and Fault Messages

Table C–62 lists the settings for the Request, Response, and Fault messages. You configure these settings for message signing and encryption.

Table C–62 Request, Response, and Fault Message Signing and Encryption Settings

Name	Description	Default Value
Include Entire Body	Sign or encrypt the entire body of the SOAP message. If false, you can add specific body elements using the Body Elements section.	True for Request and Response messages False for Fault messages

Table C–62 (Cont.) Request, Response, and Fault Message Signing and Encryption Settings

Name	Description	Default Value
Include Attachment	Sign or encrypt SOAP messages with attachments. **Note**: This field is not applicable to MTOM attachments.	False
Include Attachment with MIME Headers	Sign or encrypt SOAP attachments with MIME headers. **Note**: This field is applicable if Include Attachment is enabled. It is not applicable to MTOM attachments.	False
Header Elements	Sign or encrypt the specified SOAP header elements. To add a header element: 1. Click **Add**. 2. Enter the namespace URI. 3. Enter the local name for the header element. 4. Click **OK**. To edit a header element: 1. Select the header element that you want to edit in the Header Elements list. 2. Click **Edit**. 3. Modify the values, as required. 4. Click **OK**. To delete a header element: 1. Select the header element that you want to delete in the Header Elements list. 2. Click **Delete**. 3. When prompted to confirm, click **OK**.	None
Body Elements	**Note**: This field is available if Include Entire Body is disabled. Sign or encrypt the specified body elements. This field is applicable if the Include Body field is disabled. To add a body element: 1. Click **Add**. 2. Enter the namespace URI. 3. Enter the local name for the body element. 4. Click **OK**. To edit a body element: 1. Select the bpdu element that you want to edit in the Body Elements list. 2. Click **Edit**. 3. Modify the values, as required. 4. Click **OK**. To delete a body element: 1. Select the body element that you want to delete in the Body Elements list. 2. Click **Delete**. 3. When prompted to confirm, click **OK**.	None

D
Schema Reference for Predefined Assertions

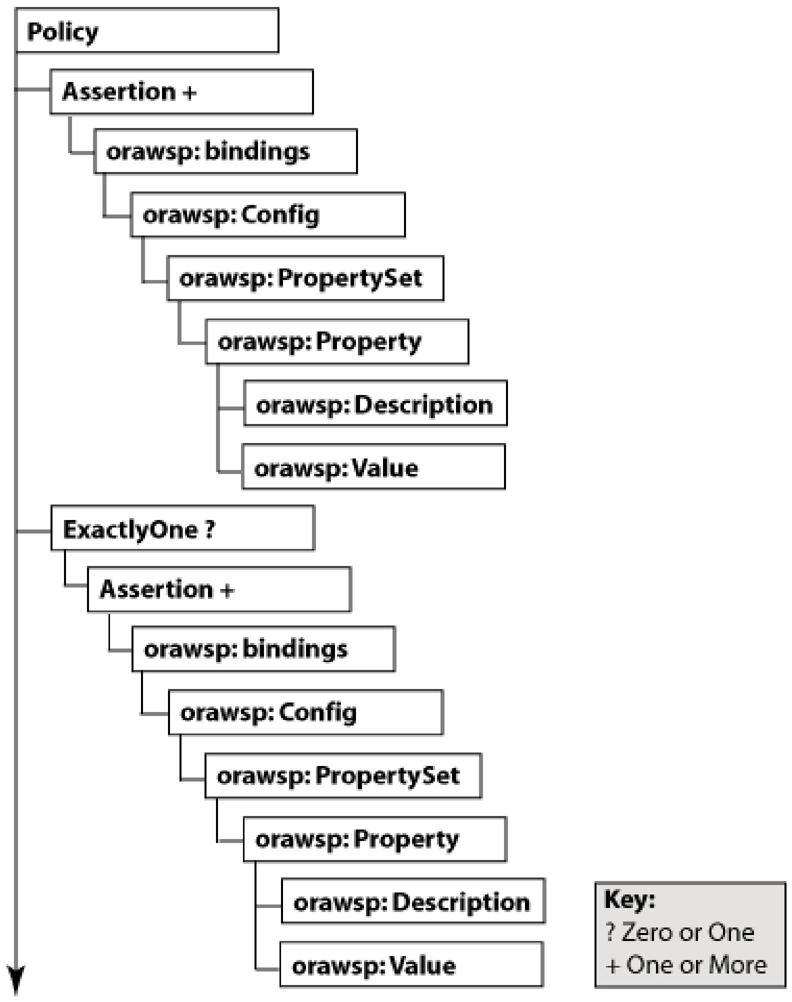
This appendix provides the XML schema for reference when creating a WS-Policy file that contains Web service assertions. Sections include:

- Graphical Representation
- Element Descriptions

Graphical Representation

The following graphic describes the element hierarchy of the assertions in the WS-Policy file.

Figure D–1 Element Hierarchy of Custom Assertion



The following sections describe each element and their subelements in detail:

- wsp:Policy
- wsp:ExactlyOne
- orasp:Assertion
- orawsp:bindings
- orawsp:Config
- orawsp:PropertySet
- orawsp:Property
- orawsp:Description
- orawsp:Value

Element Descriptions

The following sections describe the elements in the assertion in more detail. The main elements are described up front. The subelements are described following the main elements and are organized in alphabetical order.

wsp:Policy

Groups nested policy assertions.

Attributes

The following table summarizes the WS-Policy attributes, including the Oracle extensions.

Table D–1 Oracle Extensions to WS-Policy Attributes

Attribute	Description
Name	Name of the policy.
attachTo	Policy subjects to which the policy can be attached. Valid values include:binding.client, binding.server, binding.any.
category	Category of the policy. Valid values include: security, mtom, wsrm, addressing, and management.
description	Description of the policy.
displayName	Name displayed in the user interface.
localOptimization	Flag that specifies whether local optimization is enabled. Oracle WSM supports a SOA local optimization feature for composite-to-composite invocations in which the reference of one composite specifies a Web service binding to a second composite. Valid values include: ■ On—Local optimization is enabled ■ Off—Local optimization is turned off. The request goes through the usual WS/SOAP/HTTP process ■ Check Identity—Optimize only if a JAAS subject already exists in the current thread, indicating that authentication has already succeeded. Otherwise, go through the usual WS/SOAP/HTTP process.
status	Status of the policy reference. Valid values include: enabled and disabled.
smartDigest	Smart Digest.
oraSmartDigest	Smart Digest.
subjectCount	Number of subjects to which the policy is attached currently.
versionCreator	Author of the current version.
versionNumber	Number of the current version.
versionTime	Time the current version was creatd.
id	Policy ID.

Example

```
<wsp:Policy
 xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
 xmlns="http://schemas.xmlsoap.org/ws/2004/09/policy"
 xmlns:oralgp="http://schemas.oracle.com/ws/2006/01/loggingpolicy"
 xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
 xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-util
ity-1.0.xsd"
 Name="oracle/wss11_x509_token_with_message_protection_client_policy"
 orawsp:attachTo="binding.client"
 orawsp:category="security"
orawsp:description="i18n:oracle.wsm.resources.policydescription.PolicyDescription
Bundle_oracle/wss11_x509_token_with_message_protection_client_policy_PolyDescKey"
orawsp:displayName="i18n:oracle.wsm.resources.policydescription.PolicyDescription
Bundle_oracle/wss11_x509_token_with_message_protection_client_policy_
PolyDispNameKey"
 orawsp:local-optimization="check-identity"
 orawsp:oraSmartDigest="935231872"
 orawsp:smartDigest="201244603"
 orawsp:status="enabled"
 orawsp:versionCreator="mdsInternal"
 orawsp:versionNumber="1"
 orawsp:versionTime="1238006529607"
 wsu:Id="wss11_x509_token_with_message_protection_client_policy">
...
</wsp:Policy>
```

wsp:ExactlyOne

Optional element that defines an OR group. For more information about OR groups, see "Defining Multiple Policy Alternatives (OR Groups)" on page 3-8.

Attributes

The following table summarizes the attribute of the <wsp:ExactlyOne> element.

Table D–2 Attribute of <wsp:ExactlyOne> Element

Attribute	Description
Name	Set to OR to indicate that this is an OR group.

Example

```
<wsp:ExactlyOne orawsp:name="Or">
<orasp:wss11-saml-with-certificates orawsp:Enforced="true" orawsp:Silent="false"
   orawsp:category="security/msg-protection, security/authentication"
   orawsp:name="WS-Security 1.1 Saml  with certificates">
<orasp:saml-token orasp:confirmation-type="sender-vouches"
   orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
<orasp:x509-token orasp:enc-key-ref-mech="thumbprint" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:sign-key-ref-mech="direct"/>
<orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:confirm-signature="true" orasp:encrypt-signature="false"
   orasp:include-timestamp="true" orasp:sign-then-encrypt="true"
   orasp:use-derived-keys="false">
...
<orasp:wss11-username-with-certificates orawsp:Enforced="true"
   orawsp:Silent="false" orawsp:category="security/authentication,
   security/msg-protection"
   orawsp:name="WS-Security 1.1 username with certificates">
<orasp:username-token orasp:add-created="false" orasp:add-nonce="false"
   orasp:is-encrypted="true" orasp:is-signed="true"
   orasp:password-type="plaintext"/>
<orasp:x509-token orasp:enc-key-ref-mech="thumbprint"
   orasp:is-encrypted="false" orasp:is-signed="true"
   orasp:sign-key-ref-mech="thumbprint"/>
<orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:confirm-signature="true" orasp:encrypt-signature="false"
   orasp:include-timestamp="true" orasp:sign-then-encrypt="true"
   orasp:use-derived-keys="false">
...
</wsp:ExactlyOne>
```

orasp:Assertion

Main element of the assertion. Valid assertion elements include:

- oralgp:Logging
- orasp:binding-authorization
- orasp:binding-permission-authorization
- orasp:coreid-security
- orasp:http-security
- orasp:kerberos-security
- orasp:sca-component-authorization
- orasp:sca-component-permission-authorization
- orasp:wss10-anonymous-with-certificates
- orasp:wss10-mutual-auth-with-certificates
- orasp:wss10-saml-hok-with-certificates
- orasp:wss10-saml-token
- orasp:wss10-saml-with-certificates
- orasp:wss10-username-with-certificates
- orasp:wss11-anonymous-with-certificates
- orasp:wss11-mutual-auth-with-certificates
- orasp:wss11-saml-with-certificates
- orasp:wss11-username-with-certificates
- orasp:wss-saml-token-bearer-over-ssl
- orasp:wss-saml-token-over-ssl
- orasp:wss-username-token
- orasp:wss-username-token-over-ssl
- rm:RMAssertion
- wsaw:UsingAddressing
- wsoma:OptimizedMimeSerialization

Attributes

The following table summarizes the attributes of the <orasp:Assertion> element.

Table D–3 Attributes of <orasp:Assertion> Element

Attribute	Description
Optional	Flag that specifies whether the assertion is optional or required.

Table D–3 (Cont.) Attributes of <orasp:Assertion> Element

Attribute	Description
Silent	Flag that specifies whether the assertion is advertised. If set to true, the assertion is not advertised.
Enforced	Flag that specifies whether the assertion is currently enabled. Valid values are true or false.
name	Name of the assertion.
description	Description of the assertion.
category	Category to which the assertion applies. Valid values include: security/authentication, security/msg-protection, security/authorization, security/logging, mtom, wsrm, addressing, and management.

Example

```
<orasp:wss11-mutual-auth-with-certificates orawsp:Enforced="true"
  orawsp:Silent="false" orawsp:category="security/authentication,
  security/msg-protection"
  orawsp:name="WS-Security 1.1 Mutual Auth with certificates">
...
</orasp:wss11-mutual-auth-with-certificates>
```

orawsp:bindings

The <oraswsp:bindings> element defines the bindings in the assertion. This element contains the following subelement:

- orawsp:Config

Example

```
<orawsp:bindings>
  <orawsp:Config orawsp:configType="declarative"
   orawsp:name="Wss11SamlWithCertsConfig">
    <orawsp:PropertySet orawsp:name="standard-security-properties">
      <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
       orawsp:type="string">
        <orawsp:Value>ultimateReceiver</orawsp:Value>
      </orawsp:Property>
    </orawsp:PropertySet>
  </orawsp:Config>
 </orawsp:bindings>
```

orawsp:Config

The <oraswsp:Config> element defines the configuration for the assertion. This element can contain the following subelement:

- orawsp:PropertySet

Attributes

The following table summarizes the attributes of the <orawsp:Config> element.

Table D–4 Attributes of <orawsp:Config> Element

Attribute	Description
name	Name of the configuration.
type	Category to which the configuration applies.
configType	Configuration type. Valid values include: declarative and programmatic. ■ declarative—Use deployment descriptors and configuration files to describe authentication and authorization requirements. ■ programmatic—Embed security enforcement within the application.

Example

```
<orawsp:Config orawsp:configType="declarative"
 orawsp:name="Wss11SamlWithCertsConfig">
  <orawsp:PropertySet orawsp:name="standard-security-properties">
    <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
     orawsp:type="string">
      <orawsp:Value>ultimateReceiver</orawsp:Value>
    </orawsp:Property>
  </orawsp:PropertySet>
</orawsp:Config>
```

orawsp:PropertySet

The <oraswsp:PropertySet> element groups nested properties. This element contains the following subelement:

- orawsp:Property

Attributes

The following table summarizes the attributes of the <orawsp:PropertySet> element.

Table D–5 Attributes of <orawsp:PropertySet> Element

Attribute	Description
name	Name of the property set.

Example

```
  <orawsp:PropertySet orawsp:name="standard-security-properties">
    <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
     orawsp:type="string">
      <orawsp:Value>ultimateReceiver</orawsp:Value>
    </orawsp:Property>
  </orawsp:PropertySet>
```

orawsp:Property

The <oraswsp:Property> element defines a single property. The following summarize valid properties used by the predefined assertions.

The <orawsp:Property> element can contain the following subelements:

- orawsp:Value

Attributes

The following table summarizes the attributes of the <orawsp:Property> element.

Table D–6 Attributes of <orawsp:Property> Element

Attribute	Description
name	Name of the property. See Table D–7 for a list of property values used by the predefined assertions.
type	Type of the property. For example, string.
contentType	Specifies whether the property is required and can be overridden. Valid values include: ■ constant—Property is a constant value and cannot be overridden. ■ required—Property is required and can be overridden. ■ optional—Property is optional and can be overridden. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13.

The following table summarizes the properties used by the predefined assertions.

Table D–7 Properties Used by the Predefined Assertions

Property	Description
action	Action or Web service operation for which authorization checks are performed. This value can be a comma-separated list of values. This field accepts wildcards. For example, `validate,amountAvailable`.
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.

Table D–7 (Cont.) Properties Used by the Predefined Assertions

Property	**Description**
DeliveryAssurance	Delivery assurance. Valid values include: ▪ InOrder—Messages are delivered in the order they were sent. This is the default. ▪ AtLeastOnce—Every message is delivered at least once. It is possible that some messages are delivered more than once. ▪ AtLeastOnceInOrder—Every message is delivered at least once and in the order they were sent. It is possible that some messages are delivered more than once. ▪ ExactlyOnce—Every message is delivered exactly once, without duplication. ▪ ExactlyOnceInOrder—Every message is delivered exactly once, without duplication, and in the order they were sent. ▪ AtMostOnce—Messages are delivered at most once, without duplication. It is possible that some messages may not be delivered at all. ▪ AtMostOnceInOrder—Messages are delivered at most once, without duplication and in the order received. It is possible that some messages may not be delivered at all.
jdbc-connection-name	JNDI reference to a JDBC data store. Valid when the StoreType is set to JDBC. This value defaults to jdbc/MessagesStore.
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.recipient.alias	Keystore alias associated with the peer certificate. The security runtime uses this alias to extract the peer certificate from the configured keystore and to encrypt messages to the peer. Can be superseded by "Using Service Identity Certification Extension".
permission-class	Class used for the permission-based checking. For example, `oracle.wsm.security.WSFuncPermission`.
realm	HTTP realm. This value defaults to owsm.
resource	Name of the resource for which authorization checks are performed. This field accepts wildcards. For example, if the namespace of the Web service is `http://project11` and the service name is `CreditValidation`, the resource name is `http://project11/CreditValidation`.
role	SOAP role. This value defaults to ultimateReceiver.
saml.assertion.filename	File containing SAML assertions. This value defaults to temp.
saml.issuer.name	Name of the issuer of the SAML token. This value defaults to www.oracle.com.
StoreName	Name of the message store. This value defaults to oracle.

Table D–7 (Cont.) Properties Used by the Predefined Assertions

Property	Description
StoreType	Type of message store. Valid values include: ■ InMemory—Messages are stored in memory. This is the default. ■ JDBC—Messages are stored using JDBC.
user.roles.include	SOAP roles to be included. This value defaults to false.

Example

```
<orawsp:PropertySet orawsp:name="standard-security-properties">
  <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
   orawsp:type="string">
    <orawsp:Value>ultimateReceiver</orawsp:Value>
  </orawsp:Property>
</orawsp:PropertySet>
```

orawsp:Description

The <oraswsp:Description> element provides a description of the property.

Example

```
<orawsp:Description>My description.</orawsp:Description>
```

orawsp:Value

The <oraswsp:Value> element provides a list of valid values for the property.

Example

```
<orawsp:Value>ultimateReceiver</orawsp:Value>
```

oralgp:Logging

The <orasp:Logging> element defines the logging policy.

The <orasp:Logging> element contains the following subelements:

- oralgp:msg-log
- orawsp:bindings

Example

```
<oralgp:Logging orawsp:Enforced="false" orawsp:Silent="true"
 orawsp:category="security/logging" orawsp:name="Log Message1">
  <oralgp:msg-log>
    <oralgp:request>all</oralgp:request>
    <oralgp:response>all</oralgp:response>
    <oralgp:fault>all</oralgp:fault>
  </oralgp:msg-log>
  <orawsp:bindings>
    <orawsp:Config orawsp:name="added-from-em"/>
  </orawsp:bindings>
</oralgp:Logging>
```

orasp:binding-authorization

The <orasp:binding-authorization> element defines a simple role-based authorization for the request based on the authenticated subject at the SOAP binding level.

The <orasp:binding-authorization> element contains the following subelement:

- orawsp:bindings

It also contains **one** of the following subelements:

- orasp:denyAll
- orasp:permitAll
- orasp:role

Example

```
<orasp:binding-authorization orawsp:Enforced="true" orawsp:Silent="true"
 orawsp:category="security/authorization"
 orawsp:name="J2EE services Authorization">
  <orasp:denyAll/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative" orawsp:name="AuthzConfig"/>
  </orawsp:bindings>
</orasp:binding-authorization>
```

orasp:binding-permission-authorization

The <orasp:binding-permission-authorization> element defines simple permission-based authorization for the request based on the authenticated subject at the SOAP binding level.

The <orasp:binding-permission-authorization> element contains the following subelements:

- orasp:check-permission
- orawsp:bindings
- orawsp:guard

Example

```
<orasp:binding-permission-authorization orawsp:Enforced="true"
 orawsp:Silent="true" orawsp:category="security/authorization"
 orawsp:name="J2EE Permission Based Authorization">
  <orasp:check-permission/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="BindingPermissionAuthzConfig">
      <orawsp:PropertySet orawsp:name="perms-authz-properties">
        <orawsp:Property orawsp:contentType="optional" orawsp:name="resource"
         orawsp:type="string">
          <orawsp:DefaultValue>*</orawsp:DefaultValue>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional" orawsp:name="action"
         orawsp:type="string">
          <orawsp:DefaultValue>*</orawsp:DefaultValue>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="permission-class" orawsp:type="string">
          <orawsp:DefaultValue>oracle.wsm.security.WSFunctionPermission
          </orawsp:DefaultValue>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
  <orawsp:guard>
    <orawsp:resource-match>*</orawsp:resource-match>
    <orawsp:action-match>*</orawsp:action-match>
  </orawsp:guard>
 </orasp:binding-permission-authorization>
```

orasp:coreid-security

The <orasp:coreid-security> element uses the credentials in the WS-Security header's binary security token to authenticate users against the Oracle Access Manager identity store.

It contains the following subelements:

- orasp:coreid-token
- orawsp:bindings

Example

```
<orasp:coreid-security orawsp:Enforced="true" orawsp:Silent="true"
 orawsp:category="security/authentication, security/authorization"
 orawsp:name="OAM Security">
  <orasp:coreid-token orasp:is-encrypted="false" orasp:is-signed="false"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative" orawsp:name="CoreIdConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
       </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:coreid-security>
```

orasp:http-security

The <orasp:http-security> element uses the credentials in the HTTP header to authenticate users against the Oracle Platform Security Services identity store.

It contains the following subelements:

- orasp:auth-header
- orasp:require-tls
- orawsp:bindings

Example

```
<orasp:http-security orawsp:Enforced="true" orawsp:Silent="true"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="Http over SSL Security">
  <orasp:auth-header orasp:mechanism="basic"/>
  <orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative" orawsp:name="HttpConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="realm"
         orawsp:type="string">
          <orawsp:Value>owsm</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:http-security>
```

orasp:kerberos-security

The <orasp:kerberos-security> element enforces in accordance with the WS-Security Kerberos Token Profile v1.1 standard.

It contains the following subelements:

- orasp:kerberos-token
- orawsp:bindings
- orasp:msg-security

Example

```
<orasp:kerberos-security orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication" orawsp:name="WSS Kerberos Token">
  <orasp:kerberos-token orasp:is-encrypted="false" orasp:is-signed="false"
   orasp:type="gss-apreq-v5"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="KerberosSecurityConfig"/>
  </orawsp:bindings>
</orasp:kerberos-security>
```

orasp:sca-component-authorization

The <orasp:sca-component-authorization> element defines simple role-based authorization for the request based on the authenticated subject at the SOA component level.

The <orasp:sca-component-authorization> element contains the following subelement:

- orawsp:bindings

It also contains **one** of the following subelements:

- orasp:denyAll

- orasp:permitAll
- orasp:role

Example

```
<orasp:sca-component-authorization orawsp:Enforced="true" orawsp:Silent="true"
 orawsp:category="security/authorization" orawsp:name="Fabric Component
 Authorization">
  <orasp:denyAll/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="FabricAuthzConfig"/>
  </orawsp:bindings>
 </orasp:sca-component-authorization>
```

orasp:sca-component-permission-authorization

The <orasp:sca-component-permission-authorization> element provides simple permission-based authorization for the request based on the authenticated subject at the SOA component level.

The <orasp:binding-permission-authorization> element contains the following subelements:

- orasp:check-permission
- orawsp:bindings
- orawsp:guard

Example

```
<orasp:sca-component-permission-authorization orawsp:Enforced="true"
 orawsp:Silent="true" orawsp:category="security/authorization"
 orawsp:name="Fabric Component Authorization">
  <orasp:check-permission/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="FabricAuthzConfig">
      <orawsp:PropertySet orawsp:name="perms-authz-properties">
        <orawsp:Property orawsp:contentType="optional" orawsp:name="resource"
         orawsp:type="string">
          <orawsp:DefaultValue>*</orawsp:DefaultValue>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional" orawsp:name="action"
         orawsp:type="string">
         <orawsp:DefaultValue>*</orawsp:DefaultValue>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="permission-class" orawsp:type="string">
          <orawsp:DefaultValue>
         oracle.wsm.security.WSFunctionPermission</orawsp:DefaultValue>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
  <orawsp:guard>
    <orawsp:resource-match>*</orawsp:resource-match>
    <orawsp:action-match>*</orawsp:action-match>
  </orawsp:guard>
```

```
</orasp:sca-component-permission-authorization>
```

orasp:wss10-anonymous-with-certificates

The <orasp:wss10-anonymous-with-certificates> element provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.0 standard.

It contains the following subelements:

- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

Example

```
<orasp:wss10-anonymous-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false" orawsp:category="security/msg-protection"
orawsp:name="WS-Security 1.0 Anonymous with certificates">
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss10AnonWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss10-anonymous-with-certificates>
```

orasp:wss10-mutual-auth-with-certificates

The <orasp:wss10-mutual-auth-with-certificates> element enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

It contains the following subelements:

- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

Example

```
<orasp:wss10-mutual-auth-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false" orawsp:category="security/authentication,
 security/msg-protection" orawsp:name="WS-Security 1.0 Mutual Auth with
 certificates">
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss10AnonWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss10-mutual-auth-with-certificates>
```

orasp:wss10-saml-hok-with-certificates

The <orasp:wss1-saml-hok-with-certificates> element provides message protection (integrity and confidentiality) and SAML holder of key based authentication for outbound SOAP messages in accordance with the WS-Security 1.0 standard.

It contains the following subelements:

- orasp:saml-token
- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

Example

```
<orasp:wss10-saml-hok-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false" orawsp:category="security/authentication,
 security/msg-protection" orawsp:name="WS-Security 1.0 SAML Holder Of Key
 with certificates">
  <orasp:saml-token orasp:confirmation-type="holder-of-key"
   orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
  <orasp:x509-token orasp:enc-key-ref-mech="direct"
   orasp:is-encrypted="false" orasp:is-signed="true"
   orasp:rcpt-enc-key-ref-mech="direct" orasp:rcpt-sign-key-ref-mech="direct"
   orasp:sign-key-ref-mech="ski"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
        <orasp:encrypted-parts>
          <orasp:body/>
        </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss10SamlHOKWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:name="keystore.recipient.alias"
         orawsp:type="string">
          <orawsp:Value>orakey</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="saml.issuer.name" orawsp:type="string">
          <orawsp:Value>www.oracle.com</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
```

```
         orawsp:name="user.roles.include" orawsp:type="string">
          <orawsp:Value>false</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
          orawsp:name="saml.assertion.filename" orawsp:type="string">
          <orawsp:Value>temp</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss10-saml-hok-with-certificates>
```

orasp:wss10-saml-token

The <orasp:wss10-saml-token> element authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header.

It contains the following subelements:

- orasp:saml-token
- orawsp:bindings

Example

```
<orasp:wss10-saml-token orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication" orawsp:name="WSSecurity SAML Token">
  <orasp:saml-token orasp:confirmation-type="sender-vouches"
   orasp:is-encrypted="false" orasp:is-signed="false" orasp:version="1.1"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="WssSamlTokenConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
           orawsp:type="string">
            <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss10-saml-token>
```

orasp:wss10-saml-with-certificates

The <orasp:wss10-saml-with-certificates> element enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

It contains the following subelements:

- orasp:saml-token
- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

Example

```
<orasp:wss10-saml-with-certificates orawsp:Enforced="true"
```

```
 orawsp:Silent="false" orawsp:category="security/authentication,
 security/msg-protection" orawsp:name="WS-Security 1.0 SAML with certificates">
  <orasp:saml-token orasp:confirmation-type="sender-vouches"
   orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss10SamlWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss10-saml-with-certificates>
```

orasp:wss10-username-with-certificates

The <orasp:wss10-username-with-certificates> element enforces message protection (integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard.

It contains the following subelements:

- orasp:username-token
- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

Example

```
<orasp:wss10-username-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="WS-Security 1.0 username with certificates">
  <orasp:username-token orasp:add-created="false" orasp:add-nonce="false"
   orasp:is-encrypted="true" orasp:is-signed="true"
   orasp:password-type="plaintext"/>
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
   </orasp:response>
   <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss10UsernameWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss10-username-with-certificates>
```

orasp:wss11-anonymous-with-certificates

The <orasp:wss11-anonymous-with-certificates> element provides message protection (integrity and confidentiality) for outbound SOAP requests in accordance with the WS-Security 1.1 standard.

It contains the following subelements:

- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

Example

```
<orasp:wss11-anonymous-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false" orawsp:category="security/msg-protection"
 orawsp:name="WS-Security 1.0 Anonymous with certificates">
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss11AnonWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss11-anonymous-with-certificates>
```

orasp:wss11-mutual-auth-with-certificates

The <orasp:wss11-mutual-auth-with-certificates> element enforces message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

It contains the following subelements:

- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

Example

```
<orasp:wss11-mutual-auth-with-certificates orawsp:Enforced="true"
  orawsp:Silent="false" orawsp:category="security/authentication,
  security/msg-protection"
  orawsp:name="WS-Security 1.1 Mutual Auth with certificates">
  <orasp:x509-token orasp:enc-key-ref-mech="thumbprint"
   orasp:is-encrypted="false" orasp:is-signed="true"
```

```
    orasp:sign-key-ref-mech="direct"/>
   <orasp:msg-security orasp:algorithm-suite="Basic128"
    orasp:confirm-signature="false" orasp:encrypt-signature="false"
    orasp:include-timestamp="true" orasp:sign-then-encrypt="true"
    orasp:use-derived-keys="false">
     <orasp:request>
       <orasp:signed-parts>
         <orasp:body/>
       </orasp:signed-parts>
       <orasp:encrypted-parts>
         <orasp:body/>
       </orasp:encrypted-parts>
     </orasp:request>
     <orasp:response>
       <orasp:signed-parts>
         <orasp:body/>
       </orasp:signed-parts>
       <orasp:encrypted-parts>
         <orasp:body/>
       </orasp:encrypted-parts>
     </orasp:response>
     <orasp:fault/>
   </orasp:msg-security>
   <orawsp:bindings>
     <orawsp:Config orawsp:configType="declarative"
      orawsp:name="Wss10AnonWithCertsConfig">
       <orawsp:PropertySet orawsp:name="standard-security-properties">
         <orawsp:Property orawsp:name="keystore.recipient.alias"
          orawsp:type="string">
            <orawsp:Value>orakey</orawsp:Value>
         </orawsp:Property>
      </orawsp:PropertySet>
     </orawsp:Config>
   </orawsp:bindings>
</orasp:wss11-mutual-auth-with-certificates>
```

orasp:wss11-saml-with-certificates

The <orasp:wss11-saml-with-certificates> element enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

It contains the following subelements:

- orasp:saml-token
- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

Example

```
<orasp:wss11-saml-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false" orawsp:category="security/authentication,
 security/msg-protection" orawsp:name="WS-Security 1.1 SAML with certificates">
  <orasp:saml-token orasp:confirmation-type="sender-vouches"
   orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:response>
    <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss11SamlWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss11-saml-with-certificates>
```

orasp:wss11-username-with-certificates

The <orasp:wss11-username-with-certificates> element enforces message protection (integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.

It contains the following subelements:

- orasp:username-token
- orasp:x509-token
- orasp:msg-security
- orawsp:bindings

Example

```
<orasp:wss11-username-with-certificates orawsp:Enforced="true"
 orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="WS-Security 1.1 username with certificates">
  <orasp:username-token orasp:add-created="false" orasp:add-nonce="false"
   orasp:is-encrypted="true" orasp:is-signed="true"
   orasp:password-type="plaintext"/>
  <orasp:x509-token orasp:enc-key-ref-mech="direct" orasp:is-encrypted="false"
```

```
   orasp:is-signed="true" orasp:rcpt-enc-key-ref-mech="direct"
   orasp:rcpt-sign-key-ref-mech="direct" orasp:sign-key-ref-mech="direct"/>
  <orasp:msg-security orasp:algorithm-suite="Basic128"
   orasp:encrypt-signature="false" orasp:include-timestamp="true"
   orasp:sign-then-encrypt="true">
    <orasp:request>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
    </orasp:request>
    <orasp:response>
      <orasp:signed-parts>
        <orasp:body/>
      </orasp:signed-parts>
      <orasp:encrypted-parts>
        <orasp:body/>
      </orasp:encrypted-parts>
   </orasp:response>
   <orasp:fault/>
  </orasp:msg-security>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="Wss11UsernameWithCertsConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss11-username-with-certificates>
```

orasp:wss-saml-token-bearer-over-ssl

The <orasp:wss-saml-token-bearer-over-ssl> element authenticates users using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header.

It contains the following subelements:

- orasp:saml-token
- orasp:require-tls
- orawsp:bindings

Example

```
<orasp:wss-saml-token-bearer-over-ssl orawsp:Enforced="true"
 orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="WSSecurity Saml Token With Confirmation method Bearer Over SSL ">
  <orasp:saml-token orasp:confirmation-type="bearer" orasp:is-encrypted="false"
   orasp:is-signed="false" orasp:version="1.1"/>
  <orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
  <orawsp:bindings>
```

```
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="WssSamlTokenBearerOverSSLConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="saml.issuer.name" orawsp:type="string">
          <orawsp:Value>www.oracle.com</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="user.roles.include" orawsp:type="string">
          <orawsp:Value>false</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss-saml-token-bearer-over-ssl>
```

orasp:wss-saml-token-over-ssl

The <orasp:wss-saml-token-over-ssl> element enforces the authentication of credentials provided via a SAML token within WS-Security SOAP header using the sender-vouches confirmation type.

It contains the following subelements:

- orasp:saml-token
- orasp:require-tls
- orawsp:bindings

Example

```
<orasp:wss-saml-token-over-ssl orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="WSSecurity SAML Token Over SSL">
  <orasp:saml-token orasp:confirmation-type="sender-vouches"
   orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
  <orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="true"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="WssSamlTokenOverSSLConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="saml.issuer.name" orawsp:type="string">
          <orawsp:Value>www.oracle.com</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="user.roles.include" orawsp:type="string">
          <orawsp:Value>false</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss-saml-token-over-ssl>
```

orasp:wss-username-token

The <orasp:wss-username-token> element enforces authentication with username and password credentials in the WS-Security UsernameToken SOAP header.

It contains the following subelements:

- orasp:username-token
- orawsp:bindings

Example

```
<orasp:wss-username-token orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication"
 orawsp:name="WSSecurity UserName Token">
  <orasp:username-token orasp:add-created="false" orasp:add-nonce="false"
   orasp:is-encrypted="true" orasp:is-signed="true"
   orasp:password-type="plaintext"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="WssUsernameTokenConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
</orasp:wss-username-token>
```

orasp:wss-username-token-over-ssl

The <orasp:wss-username-token-over-ssl> element uses the credentials in the UsernameToken WS-Security SOAP header to authenticate users against the Oracle Platform Security Services configured identity store.

It contains the following subelements:

- orasp:username-token
- orasp:require-tls
- orawsp:bindings

Example

```
<orasp:wss-username-token-over-ssl orawsp:Enforced="true" orawsp:Silent="false"
 orawsp:category="security/authentication, security/msg-protection"
 orawsp:name="WSSecurity UserName Token Over SSL">
  <orasp:username-token orasp:add-created="true" orasp:add-nonce="true"
   orasp:is-encrypted="true" orasp:is-signed="true"
   orasp:password-type="plaintext"/>
  <orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative"
     orawsp:name="WssUsernameTokenOverSSLConfig">
      <orawsp:PropertySet orawsp:name="standard-security-properties">
        <orawsp:Property orawsp:contentType="constant" orawsp:name="role"
         orawsp:type="string">
          <orawsp:Value>ultimateReceiver</orawsp:Value>
        </orawsp:Property>
      </orawsp:PropertySet>
    </orawsp:Config>
  </orawsp:bindings>
```

```
</orasp:wss-username-token-over-ssl>
```

rm:RMAssertion

The <rm:RMAssertion> element provides support for version 1.0 and version 1.1 of the Web Services Reliable Messaging protocol. The version supported depends on the XML schema namespace value used:

- WS-ReliableMessaging 1.1: http://docs.oasis-open.org/ws-rx/wsrmp/200702
- WS-ReliableMessaging 1.0: http://schemas.xmlsoap.org/ws/2005/02/rm/policy

This policy can be attached to any SOAP-based client or endpoint. Full support for this feature may require additional programming.

The <rm:RMAssertion> element contains the following subelement:

- orawsp:bindings

Example

```
<rm:RMAssertion xmlns:rm="http://schemas.xmlsoap.org/ws/2005/02/rm/policy"
  orawsp:Enforced="true" orawsp:Silent="false" orawsp:category="wsrm"
orawsp:description="i18n:oracle.wsm.resources.policydescription.PolicyDescriptionB
undle_oracle/wsrm10_policy_RMAssertion_AssertionDescKey"
 orawsp:name="RM 1.0">
  <wsp:Policy/>
  <orawsp:bindings>
    <orawsp:Config orawsp:name="RMConfig">
      <orawsp:PropertySet orawsp:name="standard-wsrm-properties">
        <orawsp:Property orawsp:name="DeliveryAssurance" orawsp:type="string">
          <orawsp:Description>Delivery Assurance. Possible values
           (case-insensitive) are InOrder,  AtLeastOnce, AtLeastOnceInOrder,
           ExactlyOnce, ExactlyOnceInOrder, AtMostOnce,
           AtMostOnceInOrder.</orawsp:Description>
          <orawsp:Value>inorder</orawsp:Value>
          <orawsp:DefaultValue>inorder</orawsp:DefaultValue>
        </orawsp:Property>
        <orawsp:Property orawsp:name="StoreType" orawsp:type="string">
          <orawsp:Description>The type of message store used. Possible values
           (case-insensitive) areInMemory, JDBC.</orawsp:Description>
          <orawsp:Value>inmemory</orawsp:Value>
          <orawsp:DefaultValue>inmemory</orawsp:DefaultValue>
        </orawsp:Property>
        <orawsp:Property orawsp:name="StoreName" orawsp:type="string">
          <orawsp:Description>The name of the message store.
          </orawsp:Description>
          <orawsp:Value>oracle</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:contentType="optional"
         orawsp:name="jdbc-connection-name" orawsp:type="string">
          <orawsp:Description>The JNDI reference to a JDBC data source, when
           the store type is JDBC.</orawsp:Description>
          <orawsp:Value>jdbc/MessagesStore</orawsp:Value>
        </orawsp:Property>
        <orawsp:Property orawsp:name="InactivityTimeout" orawsp:type="int">
          <orawsp:Description>The inactivity timeout duration, specified in
           milliseconds.</orawsp:Description>
          <orawsp:Value>600000</orawsp:Value>
         </orawsp:Property>
         <orawsp:Property orawsp:name="BaseRetransmissionInterval"
```

```
          orawsp:type="int">
           <orawsp:Description>The base retransmission interval, specified in
            milliseconds.</orawsp:Description>
           <orawsp:Value>3000</orawsp:Value>
         </orawsp:Property>
       </orawsp:PropertySet>
     </orawsp:Config>
  </orawsp:bindings>
</rm:RMAssertion>
```

wsaw:UsingAddressing

The <wsaw:UsingAddressing> element causes the platform to check inbound messages for the presence of WS-Addressing headers conforming to the W3C 2005 Final WS-Addressing Policy standard. In addition, it causes the platform to include a WS-Addressing header in outbound SOAP messages.

The <wsaw:UsingAddressing> element contains the following subelement:

- orawsp:bindings

Example

```
<wsaw:UsingAddressing xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
 orawsp:Enforced="true" orawsp:Silent="false" orawsp:category="addressing"
 orawsp:name="WS-Addressing 2005">
  <orawsp:bindings>
    <orawsp:Config orawsp:name="added-from-em"/>
  </orawsp:bindings>
</wsaw:UsingAddressing>
```

wsoma:OptimizedMimeSerialization

The <wsoma:OptimizedMimeSerialization> element rejects inbound messages that are not in MTOM format and verifies that outbound messages are in MTOM format. MTOM refers to specifications `http://www.w3.org/TR/2005/REC-soap12-mtom-20050125` and `http://www.w3.org/Submission/2006/SUBM-soap11mtom10-20060405` for SOAP 1.2 and SOAP 1.1 bindings, respectively.

The <wsoma:OptimizedMimeSerialization> element contains the following subelement:

- orawsp:bindings

Example

```
<wsoma:OptimizedMimeSerialization
 xmlns:wsoma=
 "http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization"
 orawsp:Enforced="true" orawsp:Silent="false" orawsp:category="mtom"
 orawsp:name="MTOM">
  <orawsp:bindings>
    <orawsp:Config orawsp:name="added-from-em"/>
  </orawsp:bindings>
</wsoma:OptimizedMimeSerialization>
```

oralgp:fault

The <oralgp:fault> element configures logging for the fault message. Valid values include:

- all—Log the entire SOAP message.
- header—Log SOAP header information only.
- soap_body—Log SOAP body information only.
- soap_envelope—Log SOAP envelope information only.

Example

```
<oralgp:msg-log>
  <oralgp:request>all</oralgp:request>
  <oralgp:response>all</oralgp:response>
  <oralgp:fault>all</oralgp:fault>
</oralgp:msg-log>
```

oralgp:request

The <oralgp:request> element configures logging for the request message. Valid values include:

- all—Log the entire SOAP message.
- header—Log SOAP header information only.
- soap_body—Log SOAP body information only.
- soap_envelope—Log SOAP envelope information only.

Example

```
<oralgp:msg-log>
  <oralgp:request>all</oralgp:request>
  <oralgp:response>all</oralgp:response>
  <oralgp:fault>all</oralgp:fault>
</oralgp:msg-log>
```

oralgp:response

The <oralgp:response> element configures logging for the response message. Valid values include:

- all—Log the entire SOAP message.
- header—Log SOAP header information only.
- soap_body—Log SOAP body information only.
- soap_envelope—Log SOAP envelope information only.

Example

```
<oralgp:msg-log>
  <oralgp:request>all</oralgp:request>
  <oralgp:response>all</oralgp:response>
  <oralgp:fault>all</oralgp:fault>
</oralgp:msg-log>
```

oralgp:msg-log

The <oralgp:msg-log> element configures logging for the request, response, and fault messages. The <oralgp:msg-log> element contains the following subelements:

- oralgp:request
- oralgp:response
- oralgp:fault

Example

```
<oralgp:msg-log>
  <oralgp:request>all</oralgp:request>
  <oralgp:response>all</oralgp:response>
  <oralgp:fault>all</oralgp:fault>
</oralgp:msg-log>
```

orasp:attachment

The <orasp:attachment> element defines the attachment information.

Attributes

The following table summarizes the attributes of the <orasp:attachment> element.

Table D–8 Attributes of <orasp:attachment> Element

Attribute	Description
include-mime-headers	Flag that specifies whether or include MIME headers. Valid values include true or false.

Example

```
<orasp:signed-parts>
  <orasp:header orasp:name="From"
    orasp:namespace="http://www.w3.org/2005/08/addressing"/>
   <orasp:attachment orasp:include-mime-headers="false"/>
</orasp:signed-parts>
```

orasp:auth-header

The <orasp:auth-header> element specifies the name of the authentication header.

Attributes

The following table summarizes the attribute of the <orasp:auth-header> element.

Table D–9 Attributes of <orasp:auth-header> Element

Attribute	Description
mechanism	Authentication mechanism. Valid values include: ■ 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.

Examples

```
<orasp:auth-header orasp:mechanism="basic"/>
```

orasp:body

The <orasp:body> element defines the message body elements that are signed and encrypted. To include the entire body, specify the body element as follows: <orasp:body/>.

Example

```
  <orasp:request>
    <orasp:signed-parts>
      <orasp:body/>
    </orasp:signed-parts>
    <orasp:encrypted-parts>
      <orasp:body/>
    </orasp:encrypted-parts>
  </orasp:request>
```

orasp:check-permission

The <orasp:check-permission> element specifies that permissions are to be checked.

Example

```
<orasp:binding-permission-authorization orawsp:Enforced="true"
 orawsp:Silent="true" orawsp:category="security/authorization"
 orawsp:name="J2EE Permission Based Authorization">
  <orasp:check-permission/>
  ...
</orasp:binding-permission-authorization>
```

orasp:coreid-token

The <orasp:coreid-token> element defines the OAM token.

Attributes

The following table summarizes the attributes of the <orasp:coreid-token> element.

Table D–10 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.

Example

```
<orasp:coreid-token orasp:is-encrypted="false" orasp:is-signed="false"/>
```

orasp:denyAll

The <orasp:denyAll> element denies all users with any roles.

Example

```
<orasp:binding-authorization orawsp:Enforced="true" orawsp:Silent="true"
 orawsp:category="security/authorization"
 orawsp:name="J2EE services Authorization">
  <orasp:denyAll/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative" orawsp:name="AuthzConfig"/>
  </orawsp:bindings>
</orasp:binding-authorization>
```

orasp:element

The <orasp:element> element defines a header or body element that is signed or encrypted.

Attributes

The following table summarizes the attributes of the <orasp:element> element.

Table D–11 Attributes of <orasp:element> Element

Attribute	Description
name	Name of the header or body element.
namespace	Namespace.

Example

```
<orasp:signed-elements>
  <orasp:element orasp:name="BodyElement"
   orasp:namespace="http://www.w3.org/2005/08/addressing">n/a</orasp:element>
</orasp:signed-elements>
```

orasp:encrypted-elements

The <orassp:encrypted-elements> element defines the message body elements that are signed. This element is valid if <orasp:encrypted-parts> is not set to <orasp:body/>

The <orassp:encrypted-parts> element contains the following subelement:

- orasp:element

Example

```
<orasp:encrypted-elements>
  <orasp:element orasp:name="Myhead"
   orasp:namespace="http://www.w3.org/2005/08/addressing">n/a</orasp:element>
</orasp:encrypted-elements>
```

orasp:encrypted-parts

The <orasp:encrypted-parts> element defines the message parts that are encrypted.

The <orasp:encrypted-parts> element contains one or more of the following subelements:

- orasp:body
- orasp:header
- orasp:attachment

Example

```
  <orasp:request>
    <orasp:signed-parts>
      <orasp:body/>
    </orasp:signed-parts>
    <orasp:encrypted-parts>
      <orasp:body/>
    </orasp:encrypted-parts>
  </orasp:request>
```

orasp:fault

The <orasp:fault> element defines the message body elements that are signed and encrypted in the fault message. The <orasp:fault> element contains the following subelements:

- orasp:signed-parts
- orasp:encrypted-parts

Example

```
  <orasp:response>
    <orasp:signed-parts>
      <orasp:body/>
    </orasp:signed-parts>
    <orasp:encrypted-parts>
      <orasp:body/>
    </orasp:encrypted-parts>
  </orasp:response>
```

orasp:header

The <orasp:header> element defines a header element.

Attributes

The following table summarizes the attributes of the <orasp:header> element.

Table D–12 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.

Example

```
<orasp:signed-parts>
  <orasp:header orasp:name="From"
    orasp:namespace="http://www.w3.org/2005/08/addressing"/>
   <orasp:attachment orasp:include-mime-headers="false"/>
</orasp:signed-parts>
```

orasp:kerberos-token

The <orasp:kerberos-token> element defines the kerberos token.

Attributes

The following table summarizes the attributes of the <orasp:kerberos-token> element.

Table D–13 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).

Example

```
<orasp:kerberos-token orasp:is-encrypted="false" orasp:is-signed="false"
 orasp:type="gss-apreq-v5"/>
```

orasp:msg-security

The <orassp:msg-security> element defines message security for the policy. You define the body elements that are signed and encrypted for the request, response, and fault.

The <orasp:msg-security> element contains the following subelements:

- orasp:request
- orasp:response
- orasp:fault

Attributes

The following table summarizes the attributes of the <orasp:msg-security> element.

Table D–14 Attributes of <orasp:msg-security> Element

Attribute	Description
algorithm-suite	Defines the algorithm suite that is used for message protection. For example, Basic128. For more information, see "Supported Algorithm Suites" on page C-54.
confirm-signature	Flag that specifies whether to send a signature confirmation back to the client. Valid values inlcude true or false.
encrypt-signature	Flag that specifies whether to send a encryption confirmation back to the client. Valid values inlcude true or false.
include-timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.
sign-then-encyrpt	Flag that specifies whether to sign the message before encrypting the message.
use-derived-keys	Flag that specifies whether to use derived keys.

Example

```
<orasp:msg-security orasp:algorithm-suite="Basic128"
orasp:confirm-signature="false" orasp:encrypt-signature="false"
orasp:include-timestamp="true" orasp:sign-then-encrypt="true"
orasp:use-derived-keys="false">
  <orasp:request>
    <orasp:signed-parts>
      <orasp:body/>
    </orasp:signed-parts>
    <orasp:encrypted-parts>
      <orasp:body/>
    </orasp:encrypted-parts>
  </orasp:request>
  <orasp:response>
    <orasp:signed-parts>
      <orasp:body/>
    </orasp:signed-parts>
    <orasp:encrypted-parts>
      <orasp:body/>
    </orasp:encrypted-parts>
  </orasp:response>
  <orasp:fault/>
</orasp:msg-security>
```

orasp:permitAll

The <orasp:permitAll> element permits all users with any roles.

Example

```
<orasp:binding-authorization orawsp:Enforced="true" orawsp:Silent="true"
 orawsp:category="security/authorization"
 orawsp:name="J2EE services Authorization">
  <orasp:permitAll/>
  <orawsp:bindings>
```

```
    <orawsp:Config orawsp:configType="declarative" orawsp:name="AuthzConfig"/>
  </orawsp:bindings>
</orasp:binding-authorization>
```

orasp:request

The <orasp:request> element defines the message body elements that are signed and encrypted in the request message. The <orasp:request> element contains the following subelements:

- orasp:signed-parts
- orasp:encrypted-parts

Example

```
  <orasp:request>
    <orasp:signed-parts>
      <orasp:body/>
    </orasp:signed-parts>
    <orasp:encrypted-parts>
      <orasp:body/>
    </orasp:encrypted-parts>
  </orasp:request>
```

orasp:require-tls

The <orasp:require-tls> element specifies whether two-way authentication is required.

Attributes

The following table summarizes the attributes of the <orasp:require-tls> element.

Table D–15 Attributes of <orawsp:require-tls> Element

Attribute	Description
include-timestamp	Flag that specifies whether to include a timestamp. A timestamp can be used to prevent replay attacks by identifying an expiration time after which the message is no longer valid.
mutual-auth	Flag that specifies whether two-way authentication is required. Valid values include true or false.

Examples

```
<orasp:require-tls orasp:include-timestamp="true" orasp:mutual-auth="false"/>
```

orawsp:resource-match

The <orawsp:resource-match> element specifies the name of the resource for which authorization checks are performed. This field accepts wildcards.

For example, if the namespace of the Web service is `http://project11` and the service name is `CreditValidation`, the resource name is `http://project11/CreditValidation`.

Examples

```
<orawsp:guard>
  <orawsp:resource-match>
    http://project11/CreditValidation
  </orawsp:resource-match>
  <orawsp:action-match>validate,amountAvailable</orawsp:action-match>
</orawsp:guard>

<orawsp:guard>
  <orawsp:resource-match>*</orawsp:resource-match>
  <orawsp:action-match>validate,amountAvailable</orawsp:action-match>
</orawsp:guard>
```

orasp:response

The <orassp:response> element defines the message body elements that are signed and encrypted in the response message. The <oraswsp:response> element contains the following subelements:

- orasp:signed-parts
- orasp:encrypted-parts

Example

```
  <orasp:response>
    <orasp:signed-parts>
      <orasp:body/>
    </orasp:signed-parts>
    <orasp:encrypted-parts>
      <orasp:body/>
    </orasp:encrypted-parts>
  </orasp:response>
```

orasp:role

The <orasp:role> element defines the roles that are permitted access.

Attribute

The following table summarizes the attribute of the <orasp:role> element.

Table D–16 Attributes of <orasp:role> Element

Attribute	Description
name	Name of the role. Valid roles include: ▪ Monitor ▪ AdminChannelUsers ▪ Administrators ▪ OracleSystemGroup ▪ Operators ▪ CrossDomainConnectors ▪ Deployers ▪ AppTesters

Example

```
<orasp:binding-authorization orawsp:Enforced="true" orawsp:Silent="true"
  orawsp:category="security/authorization" orawsp:description=""
  orawsp:name="J2EE services Authorization">
  <orasp:role orasp:name="Monitors"/>
  <orasp:role orasp:name="AdminChannelUsers"/>
  <orawsp:bindings>
    <orawsp:Config orawsp:configType="declarative" orawsp:name="AuthzConfig"/>
  </orawsp:bindings>
</orasp:binding-authorization>
```

orasp:saml-token

The <orasp:saml-token> element configures the SAML token.

Attributes

The following table summarizes the attributes of the <orasp:saml-token> element.

Table D–17 Attributes of <orasp:saml-token> Element

Attribute	Description
confirmation-type	Confirmation type. Valid values include: sender-vouches and holder-of-key. ■ sender-vouches ■ holder-of-key ■ bearer
is-encrypted	Flag that specifies whether the assertion is encrypted. Valid values include true or false.
is-signed	Flag that specifies whether the assertion is signed. Valid values include true or false.
version	SAML version. Valid values include: 1.1.

Example

```
<orasp:saml-token orasp:confirmation-type="holder-of-key"
 orasp:is-encrypted="false" orasp:is-signed="true" orasp:version="1.1"/>
```

orasp:signed-elements

The <orassp:signed-elements> element defines the message body elements that are signed. This element is valid if <orasp:signed-parts> is not set to <orasp:body/>

The <orassp:signed-elements> element contains the following subelement:

- orasp:element

Example

```
<orasp:signed-elements>
  <orasp:element orasp:name="Myhead"
   orasp:namespace="http://www.w3.org/2005/08/addressing">n/a</orasp:element>
</orasp:signed-elements>
```

orasp:signed-parts

The <orasp:signed-parts> element defines the message parts that are signed.

The <orasp:signed-parts> element contains one or more of the following subelements:

- orasp:body
- orasp:header
- orasp:attachment

Example

```
<orasp:request>
  <orasp:signed-parts>
    <orasp:body/>
  </orasp:signed-parts>
  <orasp:encrypted-parts>
    <orasp:body/>
  </orasp:encrypted-parts>
</orasp:request>
```

orasp:username-token

The <orasp:username-token> element configures the SAML token.

Attributes

The following table summarizes the attributes of the <orasp:username-token> element.

Table D–18 Attributes of <orasp:username-token> Element

Attribute	Description
add-created	Flag that specifies whether a time stamp for the creation of the username token is required. **Note**: If Password Type is set to digest, then this attribute must be set to true. Otherwise, the policy to which it is attached will not validate.
add-nonce	Flag that specifies whether a nonce must be included with the username to prevent replay attacks. **Note**: If Password Type is set to digest, then this attribute must be set to true. Otherwise, the policy to which it is attached will not validate.
is-encrypted	Flag that specifies whether the username is encrypted. Valid values include true or false.
is-signed	Flag that specifies whether the username is signed. Valid values include true or false.
password-type	Type of password required. Valid values are: ■ none—No password. ■ plaintext—Unencrypted password in clear text. ■ digest—**Not supported in this release**. Client authenticates itself by transmitting an encrypted password through the use of an MD5 digest.

Example

```
<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

The <orasp:x509-token> element defines the x.509 digital certificate.

Attributes

The following table summarizes the attributes of the <orasp:x509-token> element.

Table D–19 Attributes of <orasp:x509-token> Element

Attribute	Description
sign-key-ref-mech	Mechanism used when signing the request. Valid values include: ▪ direct—X.509 Token is included in the request. ▪ ski—Subject Key Identifier (SKI) extension value of the X.509 certificate used to reference the certificate. (Some certificates may not have this extension.) The recipient of the message looks up its keystore for a certificate corresponding to the SKI and validates the signature against it. ▪ issuerserial—Composite key of issuer name and serial number attributes used to reference the X.509 certificate. The recipient of the message looks up its keystore for a certificate corresponding to Issuer name and Serial Number and validates the signature using it. ▪ thumbprint—Fingerprint (SHA1 hash) of the contents of the certificate. Provides a method to store certificates that is low overhead. This value is valid for Encryption Key Reference Mechanism only (described below.)
enc-key-ref-mech	Mechanism used when encrypting the request. Valid values are the same as for Sign Key Reference Mechanism above.
rcpt-sign-key-ref-mech	Mechanism used when signing the receipt. Valid values are the same as for Sign Key Reference Mechanism above.
rcpt-enc-key-ref-mech	Mechanism used when encrypting the receipt. Valid values are the same as for Sign Key Reference Mechanism above.
is-encrypted	Flag that specifies whether the assertion is encrypted. Valid values include true or false.
is-signed	Flag that specifies whether the assertion is signed. Valid values include true or false.

Example

```
<orasp:x509-token orasp:enc-key-ref-mech="thumbprint"
 orasp:is-encrypted="false" orasp:is-signed="true"
 orasp:sign-key-ref-mech="direct"/>
```

orawsp:action-match

The <orawsp:resource-match> element specifies the action or Web service operation for which authorization checks are performed. This value can be a comma-separated list of values. This field accepts wildcards.

Examples

```
<orawsp:guard>
  <orawsp:resource-match>
    http://project11/CreditValidation
  </orawsp:resource-match>
  <orawsp:action-match>validate,amountAvailable</orawsp:action-match>
</orawsp:guard>

<orawsp:guard>
  <orawsp:resource-match>*</orawsp:resource-match>
  <orawsp:action-match>validate,amountAvailable</orawsp:action-match>
</orawsp:guard>
```

orawsp:Description

The <oraswsp:Description> element provides a description of the property.

Example

```
<orawsp:Description>Valid IP Values</orawsp:Description>
```

orawsp:guard

The <orawsp:guard> element defines the resource and action match values.

Examples

```
<orawsp:guard>
  <orawsp:resource-match>
    http://project11/CreditValidation
  </orawsp:resource-match>
  <orawsp:action-match>validate,amountAvailable</orawsp:action-match>
</orawsp:guard>

<orawsp:guard>
  <orawsp:resource-match>*</orawsp:resource-match>
  <orawsp:action-match>validate,amountAvailable</orawsp:action-match>
</orawsp:guard>
```

E

Schema Reference for Custom Assertions

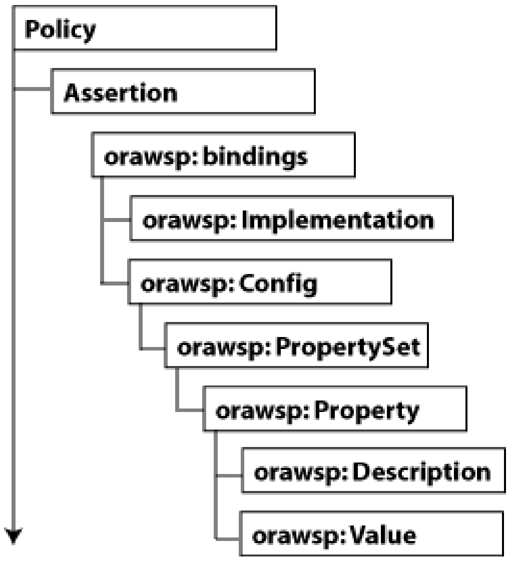
This appendix provides the XML schema for reference when creating a WS-Policy file that contains custom Web service assertions. Sections include:

- Graphical Representation
- Element Descriptions

Graphical Representation

The following graphic describes the element hierarchy of the custom assertions in the WS-Policy file.

Figure E–1 Element Hierarchy of Custom Assertion



Element Descriptions

The following sections describe the elements in the custom assertion in more detail.

wsp:Policy

Groups nested policy assertions.

Attributes

The following table summarizes the Oracle extensions to the WS-Policy attributes.

Table E–1 Oracle Extensions to WS-Policy Attributes

Attribute	Description
attachTo	Policy subjects to which the policy can be attached. Valid values include:binding.client, binding.server, binding.any.
category	Category of the policy. Valid values include: security, mtom, wsrm, addressing, and management.
description	Description of the policy.
status	Status of the policy reference. Valid values include: enabled and disabled.

Example

```
<wsp:Policy xmlns="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:orasp="http://schemas.oracle.com/ws/2006/01/securitypolicy"
  orawsp:status="enabled"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-util
ity-1.0.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  orawsp:category="security"
  orawsp:attachTo="binding.server"
  wsu:Id="ip_assertion_policy"
  xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  wsp:Name="oracle/ip_assertion_policy">
```

orasp:Assertion

Main element of the custom assertion.

Attributes

The following table summarizes the attributes of the orasp:Assertion element.

Table E–2 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.
name	Name of the assertion.
description	Description of the assertion.
category	Category to which the assertion applies. Valid values include: security/authentication, security/msg-protection, security/authorization, security/logging, mtom, wsrm, addressing, and management.

Example

```
<orasp:ipAssertion orawsp:Silent="true" orawsp:Enforced="true"
orawsp:name="WSSecurity IpAssertion Validator"
```

```
orawsp:category="security/authentication">
...
</orasp:ipAssertion>
```

orawsp:bindings

The <oraswsp:bindings> element defines the bindings in the custom assertion.

Example

```
<orawsp:bindings>
...
</orawsp:bindings>
```

orawsp:Implementation

The <oraswsp:Implementation> element defines the custom assertion implementation class.

Example

```
<orawsp:Implementation>acme.security.wss.executor.WssUsernameTokenExecutor</orawsp
:Implementation>
```

orawsp:Config

The <oraswsp:Config> element defines the configuration for the custom assertion.

Attributes

The following table summarizes the attributes of the orawsp:Config element.

Table E–3 Attributes of <orawsp:Config> Element

Attribute	Description
name	Name of the configuration.
type	Category to which the configuration applies.
configType	Configuration type. Valid values include: declarative and programmatic. ■ declarative—Use deployment descriptors and configuration files to describe authentication and authorization requirements. ■ programmatic—Embed security enforcement within the application.

Example

```
<orawsp:Config orawsp:name="ipassertion" orawsp:configType="declarative">
```

orawsp:PropertySet

The <oraswsp:PropertySet> element groups nested properties.

Attributes

The following table summarizes the attributes of the orawsp:PropertySet element.

Table E–4 Attributes of <orawsp:PropertySet> Element

Attribute	Description
name	Name of the property set.

Example

```
<orawsp:PropertySet orawsp:name="valid_ips">
```

orawsp:Property

The <oraswsp:Property> element defines a single property.

Attributes

The following table summarizes the attributes of the orawsp:Property element.

Table E–5 Attributes of <orawsp:Property> Element

Attribute	Description
name	Name of the property.
type	Type of the property. For example, string.
contentType	Specifies whether the property is required and can be overridden. Valid values include: ▪ constant—Property is a constant value and cannot be overridden. ▪ required—Property is required and can be overridden. ▪ optional—Property is optional and can be overridden. For information about overriding policies, see "Attaching Client Policies Permitting Overrides" on page 8-13.

Example

```
<orawsp:Property orawsp:name="valid_ips" orawsp:type="string"
orawsp:contentType="constant">
```

orawsp:Description

The <oraswsp:Description> element provides a description of the property.

Example

```
<orawsp:Description>Valid IP Values</orawsp:Description>
```

orawsp:Value

The <oraswsp:Value> element provides a list of valid values for the property.

Example

```
<orawsp:Value>140.87.6.143,10.178.93.107</orawsp:Value>
```