

Oracle Enterprise Gateway 11g: Security Management for SOA and Cloud

Student Guide

D73680GC10

Edition 1.0

May 2012

Dxxxxxx

ORACLE®

Author	Copyright © 2012, Oracle and/or its affiliates. All rights reserved.
Iris Li	Disclaimer
Technical Contributors and Reviewers	This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.
Gary Barg Patrice Goutin Kenneth Heung Sidharth Mishra	The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.
Editors	Restricted Rights Notice
Arijit Ghosh Smita Kommini Rashmi Rajagopal Richard Wallis	If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:
Graphic Designers	U.S. GOVERNMENT RIGHTS
Satish Bettegowda Seema Bopiah	The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.
Publishers	Trademark Notice
Jayanthy Keshavamurthy Sumesh Koshy	Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

1 Introduction

Goals 1-2
Audience 1-3
Prerequisites 1-4
Class Introductions 1-5
Course Objectives 1-6
Course Schedule 1-7
Course Map 1-8
How Can I Learn More? 1-9

2 Web Services Security Overview

Objectives 2-2
What Are Services? 2-3
Web Services Standards 2-4
Web Services Description Language (WSDL) 2-5
Defining Service Interfaces with WSDL 2-6
Security Challenges for Web Services 2-7
Quiz 2-8
Web Services Security: Concepts and Definitions 2-9
Authentication 2-10
Authorization 2-11
Confidentiality 2-12
Integrity 2-13
Identity Management 2-14
Transport Versus Message Security 2-15
Transport Security 2-16
Message Security 2-17
Quiz 2-18
XML and Web Services Security Standards 2-19
XML Signature and XML Encryption 2-20
Security Assertion Markup Language (SAML) 2-21
WS-Security: Overview 2-23
WS-Security and SAML 2-25
Best Practices for Applying Security 2-26

Quiz 2-27

Summary 2-28

3 Getting Started with Oracle Enterprise Gateway 11g

Objectives 3-2

Roadmap 3-3

Problem Statement 3-4

Oracle Solution 3-6

Basics of OEG 3-7

OEG Features 3-8

Quiz 3-9

Roadmap 3-10

OEG Architecture and Components 3-11

Enterprise Gateway Component 3-13

OEG Policy Studio 3-14

OEG Policy Center 3-15

OEG Service Explorer 3-16

OEG Service Monitor 3-18

Connections Between Components 3-19

Quiz 3-20

Roadmap 3-21

Starting the OEG Components 3-22

Connecting Policy Studio to Gateway 3-23

Oracle Enterprise Gateway Dashboard 3-25

Policy Studio Users Versus Enterprise Gateway Users 3-26

Editing Active Server Configuration 3-27

Listeners 3-29

Web Service Repository 3-30

Policies 3-31

External Connections 3-32

Certificates and Keys 3-34

Users and Groups 3-35

Service Explorer 3-36

Roadmap 3-37

Configuring HTTP Services 3-38

Organizing HTTP Services in HTTP Service Groups 3-40

Relative Path 3-42

Serving Static Content and Servlets 3-43

Adding a Remote Host 3-44

Configuring Remote Host Settings 3-45

Quiz 3-46

Summary 3-47
Lab Environment 3-48
Practice 3 Overview: Exploring the Gateway Configuration 3-49

4 Registering Web Services in OEG

Objectives 4-2
Roadmap 4-3
Services Registration 4-4
Capabilities of Registered Web Services 4-5
XML Firewalling 4-6
Enriching Messages 4-7
Dynamic Routing and Offloading 4-8
Throttling 4-9
Converting Protocol: REST and SOAP 4-10
Quiz 4-11
Roadmap 4-12
Registering a Service in Enterprise Gateway 4-13
Registering Services in Policy Studio 4-14
Steps to Register Services 4-15
What Is Created? 4-16
Service Handler 4-17
Registering Services in Service Manager 4-19
Applying Policies in Service Manager 4-20
Testing Registered Services by Using Service Explorer 4-21
Quiz 4-22
Roadmap 4-23
Policy Concepts 4-24
Global Policies 4-26
Getting Started with Policies 4-27
Filters Palette 4-28
Filters and Message Attributes 4-29
Applying Policies to Registered Web Services 4-30
Quiz 4-32
Summary 4-33
Practice 4 Overview: Registering Web Services in the Gateway 4-34

5 Monitoring, Logging, and Tracing

Objectives 5-2
Troubleshooting Tools 5-3
Roadmap 5-4
Monitoring 5-5

Enabling Monitoring 5-6
Setting Up Database for Service Monitor Reports 5-8
Configuring Database Connection for Enterprise Gateway 5-9
Monitoring Traffic 5-10
Real-Time Monitoring 5-11
Monitoring Using Service Monitor 5-12
Viewing Traffic Reports 5-13
Quiz 5-14
Roadmap 5-15
Logging Versus Tracing 5-16
Accessing Trace and Logs Online 5-17
Setting Enterprise Gateway Trace Levels 5-18
Viewing Enterprise Gateway Trace Files 5-20
Roadmap 5-21
Configuring Logging 5-22
Configuring Log Output 5-23
Configuring Log Level and Message 5-24
Log Payload Filter 5-25
Audit Trail 5-26
Configuring Audit Trail 5-27
Quiz 5-28
Summary 5-29
Practice 5 Overview: Monitoring, Logging, and Tracing 5-30

6 Managing Configurations

Objectives 6-2
Configuration Management 6-3
Configuration Structure 6-4
Storing Configuration Items 6-5
Loading Configurations 6-6
Viewing Deployed Configuration 6-7
Version Settings 6-8
Versioning Process Configuration 6-9
Versioning Configuration in Oracle Enterprise Gateway Dashboard 6-10
Versioning Configuration in Active Server Configuration 6-11
Quiz 6-12
Importing and Exporting Configurations 6-13
Exportable Configuration Items 6-14
Exporting Configuration Data 6-15
Importing Policies 6-16
Deploying the Gateway in Multiple Environments 6-17

Quiz 6-18
Summary 6-19
Practice 6 Overview: Managing Configurations 6-20

7 Fault Handling

Objectives 7-2
Abort Versus Failure 7-3
Handling Failures 7-4
SOAP Faults 7-5
SOAP Fault: Example 7-6
SOAP Fault Filter 7-7
Configuring a SOAP Fault Filter 7-8
Quiz 7-10
Example of Fault Handling 7-11
Default Fault Handler 7-12
Custom Handling for Failures 7-13
Overriding the Default Fault Handler 7-14
Custom Fault Handling by Using a Policy 7-15
Example of Custom Fault Handling: Global Handler 7-16
Example of Custom Fault Handling: Specific Handler 7-17
Common Use Case: Propagating Exceptions 7-18
Monitoring Impact 7-19
Handling Aborts 7-20
Quiz 7-21
Summary 7-22
Practice 7 Overview: Using Customized Fault Handler 7-23

8 Blocking XML Threats

Objectives 8-2
XML Concepts 8-3
XML Firewalling 8-5
XML Content and Schema Attacks 8-7
Denial of Service (DoS) 8-8
Example of XML Bomb: XML DoS DTD Recursion 8-9
Set Occurrences for Elements in XML Complexity Filter 8-10
SQL Injection 8-11
Defending Against SQL Injection 8-12
Threatening Content Filter 8-13
XML Viruses 8-14
Message Response Filtering 8-15
Schema “Poisoning” 8-16

Cryptographic Attacks 8-17
Replay Attacks 8-18
REST: Overview 8-19
Protecting REST 8-21
Summary 8-22
Practice 8 Overview: Blocking XML Attacks 8-23

9 Accelerating XML and Managing Traffic

Objectives 9-2
Roadmap 9-3
Caching: Overview 9-4
Using Caching 9-6
Configuring Caches 9-7
Configuring Local Cache Settings 9-8
Configuring Distributed Caches 9-9
Caching Response Messages: Example 9-10
Caching Filters 9-12
How Caching Works with Filters 9-13
Create Key Filter 9-14
“IsCached?” Filter 9-15
“Cache Attribute” Filter 9-16
“Removed Cached Attribute” Filter 9-17
Quiz 9-18
Roadmap 9-19
Traffic Throttling 9-20
Benefits of Throttling 9-21
How Throttling Works 9-22
Configuring a Throttling Filter 9-23
Handling Throttling Violations 9-24
Time Filter 9-25
Quiz 9-26
Summary 9-27
Practice 9 Overview 9-28

10 Configuring SSL

Objectives 10-2
SSL Basics 10-3
Digital Certificate 10-4
Chain of Trust 10-5
SSL Handshake 10-6
SSL Support in the Enterprise Gateway 10-8

Tips for Using SSL 10-9
Quiz 10-10
Configuring the HTTPS Interface 10-11
Mutual SSL Authentication 10-12
Configuring Mutual Authentication Settings 10-13
Configuring Inbound SSL Settings 10-14
Configuring Outbound SSL Settings 10-15
Quiz 10-16
Summary 10-17
Practice 10 Overview: Configuring SSL 10-18

11 Securing XML Messages

Objectives 11-2
Roadmap 11-3
Digital Signature 11-4
XML Signature 11-6
XML Signature Support in OEG 11-7
Generating XML Digital Signature 11-8
Verifying XML Digital Signature 11-9
Roadmap 11-10
XML Encryption 11-11
Basics of XML Encryption 11-12
XML Encryption: Example 11-13
XML Encryption Example 11-14
Creating Encryption Policy 11-15
XML Encryption/Decryption Filter Settings 11-16
Quiz 11-17
Roadmap 11-18
XML Message Transformation 11-19
XSLT 11-21
XSLT Transformation Filter 11-22
Manipulating HTTP Headers 11-23
Quiz 11-24
Summary 11-25
Practice 11 Overview: Securing XML Messages 11-26

12 Securing Web Services

Objectives 12-2
WS-Security: Overview 12-3
WS-Security Stack 12-5
WS-Security Architecture 12-7

Basics of WS-Security 12-8
WS-Username Token: Example 12-9
WS-Security UserNameToken Support in OEG 12-10
Validating WS-UserName Token 12-11
Inserting WS-UserName Token 12-13
Quiz 12-14
WS-Policy: Overview 12-15
Policy Assertion 12-16
Sample of a Policy File 12-17
Securing a Service By Using WS-Policy 12-18
Configuring Recipient WS-Policy 12-20
Policy Configuration Settings 12-21
Configuring Initiator WS-Policy 12-22
Quiz 12-23
Summary 12-24
Practice 12 Overview: Securing Web Services 12-25

13 Securing SOA Composites with OEG and OWSM

Objectives 13-2
SOA and Web Services 13-3
Securing SOA Composite Applications 13-4
SOA Composite Applications 13-5
SOA Composite Application: Example 13-6
Managing SOA Composite in Enterprise Manager 13-7
Oracle Service Bus 13-8
Oracle Service Bus Architecture 13-9
Virtualizing Service By Using OSB: Example 13-10
OSB Console 13-11
Quiz 13-13
Oracle Web Services Manager Policy Framework 13-14
Components of the Oracle Web Services Manager Architecture 13-15
Supported Policies 13-16
Oracle Web Services Manager Policy Assertions 13-18
Oracle WSM Predefined Policies and Assertion Templates 13-19
Applying Policies to Service in EM 13-20
Converting Authentication with OEG Policy 13-21
The Conversion Policy 13-22
Quiz 13-23
Summary 13-24
Practice 13 Overview: Securing SOA Application 13-25

14 Integrating with Identity and Access Management

- Objectives 14-2
- Oracle Access Management Suite 14-3
- Key Features of OAM 14-5
- OAM Architecture 14-6
 - Authentication By Using OEG and OAM 14-7
 - OEG Gateway as an AccessGate 14-8
 - Key Steps of Authentication via OEG and OAM 14-9
 - Creating AccessGate Entry 14-10
 - Configure OAM Authentication Repository 14-11
 - Authentication Policy 14-12
 - Quiz 14-13
 - Roadmap 14-14
 - Fine-grain Versus Coarse-Grain Authorization 14-15
 - Using OES for Authorization 14-16
 - OES Architecture 14-17
 - OAM, OES and OPSS 14-19
 - Authorization By Using OEG and OES 14-20
 - Connecting to Oracle Entitlements Server 14-21
 - Defining Authorization Policy by Using OES Admin Server 14-23
 - Authorization Policy in OES Admin Server 14-24
 - Creating Authorization Policy in OEG 14-25
 - Configuring Oracle Entitlements Server Authorization Filter 14-26
 - Quiz 14-27
 - Summary 14-28
- Demo Overview: Integrating OEG with OAM and OES 14-29

15 Securing Services in the Cloud

- Objectives 15-2
- Cloud Computing Definition 15-3
- Security Risks of Cloud Computing 15-4
- Application Programming Interface (API) and API Keys 15-5
- Security Issues Faced by Cloud Computing 15-6
- Centrally Protecting and Managing API Keys By Using OEG 15-7
- Summary 15-8

1

Introduction

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Goals

The main goals of this course are to learn to use:

- Oracle Enterprise Gateway (OEG) 11g to prevent XML-based attacks and accelerate XML
- OEG 11g along with other Oracle Fusion Middleware components to provide an end-to-end security solution for SOA composite applications and web services in the Cloud



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this course, you learn about Oracle Enterprise Gateway 11g features and functionality. You also acquire skills in using OEG tools to configure policies and filters to secure and accelerate XML and web services.

The course covers how OEG is used together with Oracle Service Bus and Oracle Web Services Manager to provide an end-to-end security solution for SOA composite applications and web services in the Cloud. You learn how OEG leverages Oracle Identity and Access Management products to provide authentication and authorization.

Audience

The target audience includes:

- Application architects
- Security developers
- Application integration specialists
- Security administrators and system administrators
- Technical personnel who are interested in securing web services, SOA, and Cloud applications



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The course is ideal for students who have a basic understanding of web application security vulnerabilities.

Prerequisites

To successfully complete this course, you should have the following:

- Understanding of XML concepts and Web Service standards such as WSDL, SOAP, and UDDI (or equivalent)
- Basic web security knowledge



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Students who have some knowledge of the concepts listed in the slide will benefit from most of the course's content.

Class Introductions

Briefly introduce yourself:

- Name
- Title or position
- Company
- Knowledge of XML and web services
- Reasons for attending this course



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Course Objectives

After completing this course, you should be able to:

- Describe the XML-based threats to an enterprise environment
- Describe web service security concepts
- Describe OEG capabilities
- Describe OEG architecture and components
- Use OEG to block XML attacks
- Use OEG to accelerate XML processing and manage traffic
- Use OEG to secure XML messages and web services
- Use OEG and OWSM to provide end-to-end security for SOA composites
- Integrate OEG with Oracle Identity Management products to provide authentication and fine-grained authorization
- Secure web services in the Cloud



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Course Schedule

Session		Module
Day 1	A.M.	1: Introduction 2: Web Services Security Overview 3: Getting Started with Oracle Enterprise Gateway 11g
	P.M.	4: Registering Web Services in OEG 5: Monitoring, Logging, and Tracing
Day 2	A.M.	6: Managing Configurations 7: Fault Handling 8: Blocking XML Threats
	P.M.	9: Accelerating XML and Managing Traffic 10: Configuring SSL
Day 3	A.M.	11: Securing XML Messages 12: Securing Web Services 13: Securing SOA Applications with OEG and OWSM
	P.M.	14: Integrating with Identity and Access Management 15: Securing Cloud Services

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Course Map

Concepts & Getting Started

Web Services Security Overview

Getting Started with OEG 11g

Registering Web Services in OEG

Management & Tools

Monitoring, Logging, Tracing

Managing Configurations

Fault Handling

Core Functions

Blocking XML Threats

Accelerating XML & Managing Traffic

Configuring SSL

Securing XML

Securing Web Services

Integration

Securing SOA Applications with OEG & OWSM

Integrating with IDM

Cloud Security with OEG

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The course map shows all the lessons of this course, and how they are grouped into logical sections.

How Can I Learn More?

Topic	Website
Education and Training	http://education.oracle.com
Product Documentation	http://www.oracle.com/technology/documentation
Product Downloads	http://www.oracle.com/technology/software
Product Articles	http://www.oracle.com/technology/pub/articles
Product Support	http://www.oracle.com/support
Product Forums	http://forums.oracle.com
Product Tutorials	http://www.otn.oracle.com/obe
Sample Code	http://www.samplecode.oracle.com/



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Web Services Security Overview

2

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to describe:

- Web services security challenges
- Key web services security concepts
- XML and web services security standards

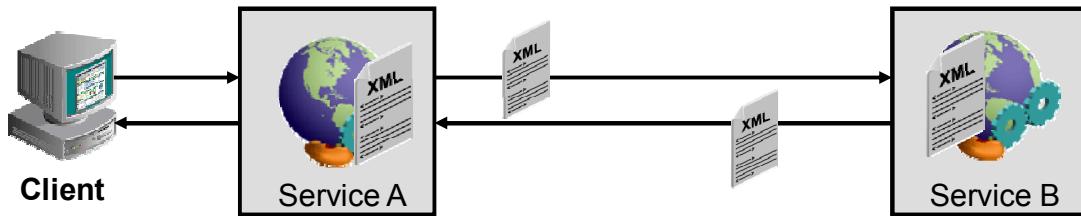


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

What Are Services?

Services are:

- IT's representation of business functionality
- Functionality-described standard interface and message structure definitions
- Accessed using standard protocols (the glue) to enable interoperability from decoupled functions



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A service is a piece of self-contained business functionality. The functionality could be as simple as storing customer data, or as complex as processing a customer's order. They perform work based on business interactions and requirements. Because services concentrate on the business value of an interface, they bridge the business/IT gap. Services interact by exchanging messages with other clients and other services.

Service functionality must be described by using standard interface and message structures to make them highly accessible and reusable. Although the IT industry has created many ways to do this, the most widely embraced are the Web Services standards.

The advantage of accessing information and services by using standard document structures, message, and protocols is that they form a consistent layer or glue between decoupled systems that enables a high degree of interoperability, regardless of the choice of operating systems and computer languages used for service implementation.

Web Services Standards

Web services:

- Rely on common standards that include:
 - Extensible Markup Language (XML) for metadata
 - Simple Object Access Protocol (SOAP): A standard format for messaging over a network
 - Web Services Description Language (WSDL): The language that provides a description for web services
 - Universal Description, Discovery, and Integration (UDDI): A web-based distributed directory to publish and locate information about web services
- Include additional specifications (WS-*) to define functionality for web services discovery, security, reliability, transactions, and management



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Web services are based on a core set of communication standards:

- **SOAP (to communicate):** Defines an XML-based structure for passing information, such as messages and data, between web services and applications
- **WSDL (to describe):** Is an XML-based language for modeling web services. A WSDL document describes the service interface, message format, and service instance location.
- **UDDI (to advertise and syndicate):** Enables organizations to register their web services in a uniform manner within a directory so that clients can locate their web services and learn how to access them

The benefit of using web services artifacts such as WSDL and XSD is that they are “accepted standards.” These XML document structures are easily exchanged using standard Internet (web services) protocols such as Hypertext Transfer Protocol (HTTP).

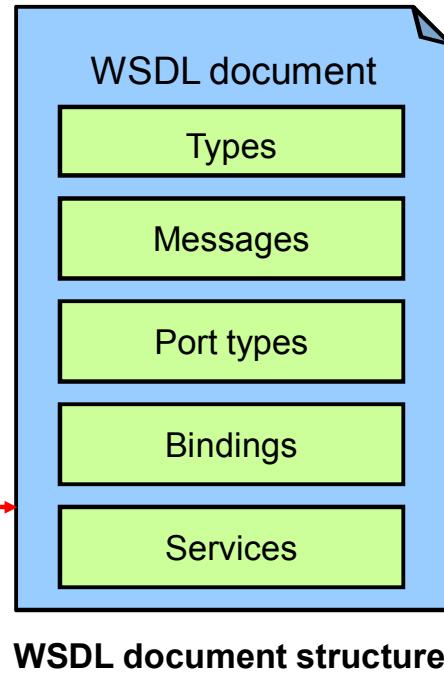
SOAP, WSDL, and UDDI are standards at the core of web services, and they are followed by many additional specifications that define reliability, security, metadata management, and transactions to meet requirements for enterprise features and qualities of service. These specifications are collectively referred to as the WS-*.

Web Services Description Language (WSDL)

- A WSDL document describes:
 - What the service does
 - How the service is accessed
 - Where the service is located
- It defines the messages and the operations of a service abstractly in XML.



Described by



WSDL document structure

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

SOAP provides a standard way of transporting messages for use by web services, whereas WSDL provides a standard way to describe web services. WSDL uses the XML format for describing the service interface with several functional (what functionality a service offers) as well as more technical aspects (how it communicates, and where it is accessible) of the contract. Because WSDL is in the standard XML format, the services developed can easily be made available to thousands of users. Users must invoke the correct services with the appropriate parameters. WSDL defines an XML grammar to specify the location of the service and to describe the operations of a service.

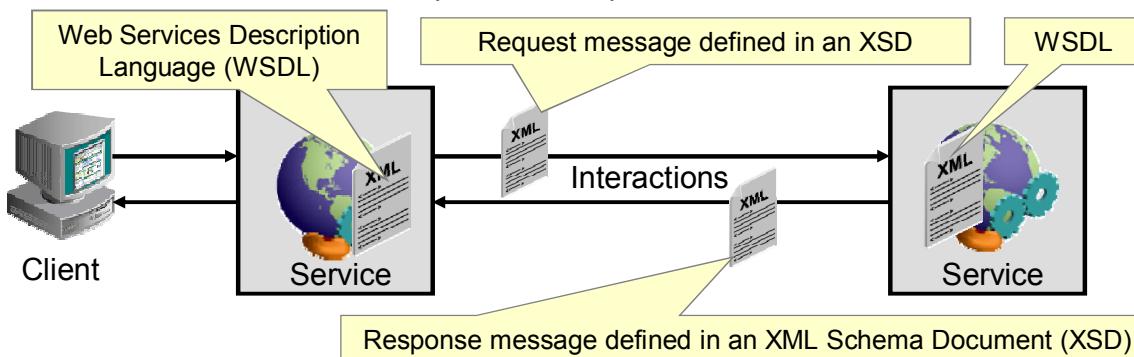
A WSDL document describes a web service by using these major elements:

- **<types>**: Defines the data types used by the Web service
- **<message>**: Describes the payload of a message used by a web service
- **<portType>**: Describes a web service, the operations that can be performed, and the messages that are involved
- **<binding>**: Defines the communication protocols used by the web service
- **<service>**: Describes the name and location of the web service and the context with which it can be accessed

Note that types, message, and portType are abstract types that are then used in bindings and services in much the same way an instance of a Java class represents a Java class.

Defining Service Interfaces with WSDL

- When defining service interfaces by using WSDL, you specify:
 - Operations that may be executed
 - Message structures for communicating the required data for each operation
- XML Schema (XSD) is used to define message structure of service interface (in WSDL).



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Before developing a service, you need to have defined the interface of that service. This is the key of the commonly accepted “WSDL first” approach to designing web services. Defining the service interface before development is important to establish a highly standardized SOA and is required to realize a number of the characteristics identified as being part of SOA.

When defining service interfaces by using WSDL, you need to specify:

- Operations that may be executed. The interface of a service defines a set of public operation signatures.
- Message structures for communicating the required data for each operation. Message structures are based on types expressed in an XML Schema (XSD) document.

XML Schema (XSD) is used to define the message structure of service interface (in WSDL). XSD is an XML language that defines and validates the structure of XML documents.

Security Challenges for Web Services

Web services:

- Are loosely coupled
- Are based on the passing of readable and self-describing business messages represented in XML
- Can easily bypass network firewalls
- Expose business functionality through open APIs
- Enable multi-hop composite applications



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Web services provide significant IT benefits, including flexibility and interoperability for data exchange and integration. But deploying networks of interconnected web services presents security challenges that include the following:

- Web services is a more loosely coupled technology compared to previous forms of middleware for distributed computing, so it needs strong security practices.
- XML-based messages can be deliberately or inadvertently malformed to break parsers or applications, thereby creating new XML threat and vulnerability protection requirements.
- Web services can be transmitted over any transport protocol, including common web protocols like HTTP. This makes it easy for web services to bypass network firewalls.
- Web services expose business functionality through open APIs, requiring new application-aware security measures.
- The XML document may participate in a multi-hop transaction or may be subject to inspection by a variety of intermediaries. This requires message-level security and auditing that can span multi-hop transactions end to end.

Quiz

The service interface is defined in the following document:

- a. SOAP
- b. XSD
- c. XSLT
- d. WSDL



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: d

Web Services Security: Concepts and Definitions

- Authentication
- Authorization
- Confidentiality
- Integrity
- Identity management
- Transport-level security
- Message-level security



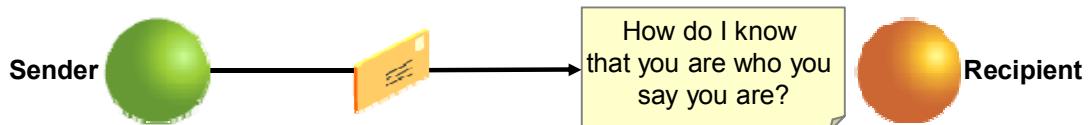
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This section discusses the basic concepts of web services and information security and the way web services security builds on existing security technology.

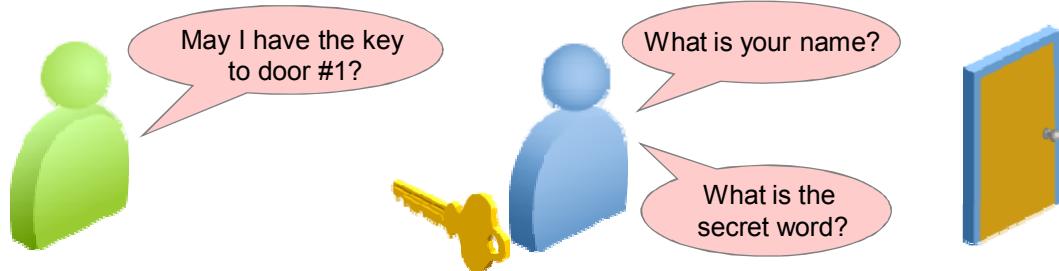
- **Authentication:** Verifying that users are who they claim to be
- **Authorization (or access control):** Granting access to specific resources based on an authenticated user's entitlements or specific role (for example, a corporate buyer)
- **Confidentiality and privacy:** Restricting access to data and messages to those identities that are allowed to view (and possibly modify) this data
- **Integrity and non-repudiation:** Ensuring that data and messages are complete, valid, and unaltered by unauthorized identities
- **Identity management:** Providing services for the retrieval and administration of identity information
- **Transport-level security:** Secures connections between the service consumer and the provider
- **Message-level security:** Secures a message throughout its journey between the sender and the intended recipient

Authentication

- Authentication is the process of verifying an identity, which can be a user, a physical device, or a service requestor.



- Authentication is generally accomplished by asking the user for an ID along with a password, token, or signature that is unique to the user and trusted to be secret.



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

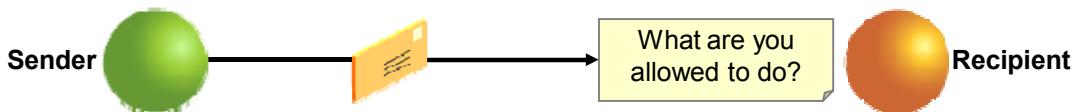
Authentication is the process of verifying that an identity is who he, she, or it claims to be. An identity can be a user, a physical device, or a service requestor. A user's identity is verified based on the credentials presented by that user, such as username and password, digital certificate, standard Security Assertion Markup Language (SAML) token, or Kerberos token. For web services, credentials are presented by a client application on behalf of the end user.

A best practice is to define a limited set of authentication levels based on the mechanisms described here. For example, basic authentication requires knowledge (username/password), medium authentication requires knowledge and possession (username/password and token), and high authentication requires verification through a biometric property.

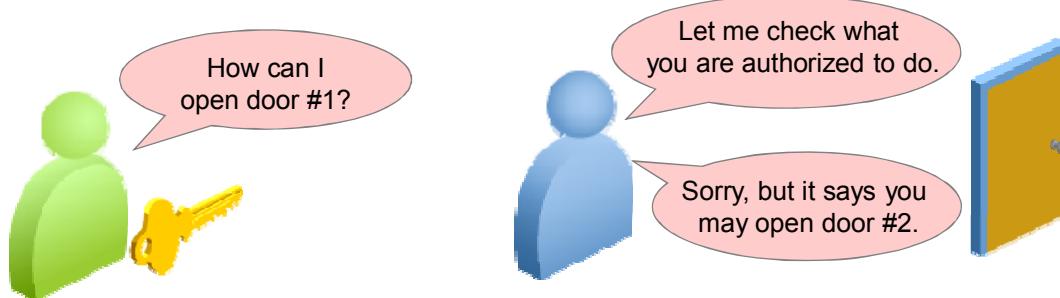
Authentication also includes single sign-on (SSO), a mechanism in which an identity needs to authenticate only once to obtain access to several IT components.

Authorization

- Authorization is the function of specifying access rights to resources.



- Authorization is generally performed by comparing rights granted to the user with operations that the user is attempting to perform.



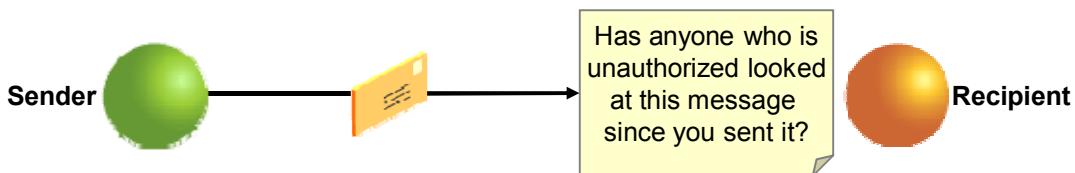
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Authorization is the function of specifying access rights to resources, which is related to information security and computer security (in general) and to access control (in particular). Authorization means determining to what extent authentication applies (in other words, what an identity is allowed to do). It grants or denies a user's operations on objects, functions, or data. Authorization should be based on the function that someone or some organization needs to do and know—no more and no less. For web services, this means checking whether the caller is allowed to call the service and/or see the result.

Confidentiality

- Confidentiality refers to the privacy of the message that has been protected throughout its message path.
- Encryption scrambles the data so that it becomes unrecognizable to everyone except those with the keys to decrypt it.
- In the case of web services, the message can be used by multiple services, so access to sensitive data should be available to only those services that need it.



ORACLE®

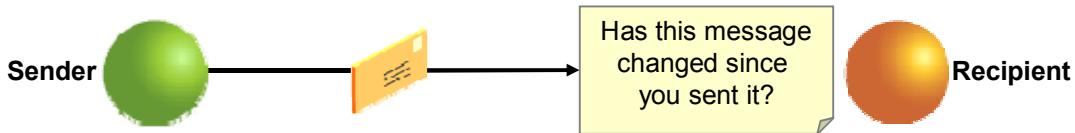
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Confidentiality means the ability to protect data from being seen by entities that it is not intended for, both while stored and in transit. Whether data remains confidential while in transit or in storage is another key aspect of security. Regarding web services, this means ensuring that no one except the service caller can see service data while it is being transferred between the provider and the requestor.

When data travels beyond the borders of a locked-down environment, it must be protected. This problem has been solved through encryption, which scrambles the data so that it becomes unrecognizable to everyone except those with the keys to decrypt it.

Integrity

- Integrity means ensuring that a message's contents have not changed during transmission.
- For web services, integrity refers to the ability to ensure that:
 - Requests were issued by the sender rather than by an imposter
 - Messages from the sender have not been altered en route
- Both of these concerns can be addressed by using digital signatures.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Integrity ensures that a message remains unaltered during transit by having an authority digitally sign that message. A digital signature also validates the sender and provides a time stamp ensuring that a transaction cannot later be repudiated by either the sender or the receiver.

XML messages are signed using the XML Signature standard.

Identity Management

- Identity management provides services for the retrieval and administration of identity information.
- Identity management includes:
 - Identity propagation: The mechanism to pass identity information within a chain of IT components
 - Identity federation: When identity propagation and administration span more than one organization



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Identity management is the administration of identities and relevant information (name, job title, password). Identities can represent both human actors (such as employees, vendors, and customers) and virtual entities (such as services and applications).

Identity management includes identity propagation. For example, a client might invoke an order process service that in turn invokes a payment service. Usually you want a service supplier to authenticate the initial service consumer: the person (or service) that started the chain of service calls, and not some intermediary component. In this example, the payment service needs to authenticate and authorize the client rather than the order process service. This implies that intermediary components between service client and provider must be able to transfer identity information such as certificates and possibly transform these from one format or protocol into another.

When identity propagation and administration span more than one organization (crossing an organization's boundary), you have *identity federation*.

Transport Versus Message Security

To secure your web service, you must configure one or both types of security:

- Transport security: Secures connections between service consumer and provider
- Message security: Secures a message throughout its journey between the sender and the intended recipient



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

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

- **Transport security:** Security begins at the transport or wire level. It allows the client and web service to communicate securely across a network by providing a secure connection.
- **Message security:** Message security is end-to-end because the message is secured throughout its journey between the sender and the intended recipient. In web service communication, the message can travel through various entities before it reaches its intended recipient. The message is secured during this journey if message-level security is applied.

Both transport and message security can be used for authentication purposes and for guaranteeing message integrity and confidentiality. You can have either kind of security—or both—configured and applied.

Transport Security

- Transport-level security secures the connection between the client and the web service with secure sockets layer (SSL).
- SSL is the most widely used transport-level data-communication protocol. SSL provides:
 - Authentication: The communication is established between two trusted parties.
 - Confidentiality: The data exchanged is encrypted.
 - Message integrity: The data is checked for possible corruption.
- Limitations:
 - Is point-to-point (secures only the connection)
 - Cannot encrypt selected parts of the message



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Transport-level security uses SSL to secure the connection between a client application and the web service. SSL provides secure connections by allowing two applications connecting over a network to authenticate each other's identity and by encrypting the data exchanged between the applications.

Transport-level security, however, secures only the connection. This means that if there is an intermediary between the client and server, such as a router or message queue, the intermediary gets the SOAP message in plain text. When the intermediary sends the message to a second receiver, the second receiver does not know who the original sender was and may not use SSL. Additionally, the encryption used by SSL is “all or nothing.” That is, either the entire SOAP message is encrypted or it is not encrypted at all. There is no way to specify that only selected parts of the message be encrypted.

Message Security

- Message-level security
 - Specifies whether the SOAP messages between a client application and the web service should be:
 - Digitally signed: Ensures data integrity
 - Encrypted: Ensures confidentiality
 - Assures authentication by requiring username, X.509, or SAML tokens
- Security data built in to the XML message header
- Advantages:
 - Provides end-to-end security
 - Allows specific parts of the message to be signed and/or encrypted



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Message security secures the message itself, often through encryption of the payload by using, for example, Public Key Infrastructure (PKI) and WS-Security.

Message-level security specifies whether SOAP messages between a client application and the web service should be digitally signed, encrypted, or both. It can also specify a shared security context between the web service and the client in the event that they exchange multiple SOAP messages. You can use message-level security to assure:

- **Confidentiality:** By encrypting message parts
- **Integrity:** By digital signatures
- **Authentication:** By requiring username, X.509, or SAML tokens

Security data is built in to the XML message text, usually as additional SOAP header fields.

Message-level security includes all the security benefits of SSL, but with additional flexibility and features. Message-level security is end-to-end, which means that a SOAP message is secure even when the transmission involves one or more intermediaries. The SOAP message itself (rather than just the connection) is digitally signed and encrypted. Finally, you can specify that only individual parts or elements of the message be signed, encrypted, or required.

Quiz

Both transport-level and message-level securities provide:

- a. Authentication
- b. Data integrity
- c. Single sign-on
- d. Data encryption



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, b, d

XML and Web Services Security Standards

Web Services Security Standards WS-Policy, WS-Security, WS-Trust, WS-Federation...
General XML-Based Security Standards SAML, XKMS, XACML ...
XML Security Standards XML Encryption, XML Signature ...
General Security Standards Kerberos, PKI, X.509, SSL ...
Security Algorithms AES, DES, RSA ...



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This table lists the security specifications for XML and web services. In principle, you can use different types of security standards as illustrated in the slide. The general security standards include well-known algorithms (such as AES and RSA) as well as basic security standards for encryption and secure conversation (such as Kerberos, SSL, PKI, and so on).

There are also special standards that deal with XML documents. Their advantage is that they read and write XML files, so the result of an encryption or signature can be processed using the usual XML processing chain. Listed first in the table are general XML-based security standards (such as SAML) and standards with special web services aspects (such as WS-Security).

The core standards include WS-Security, WS-I Basic Security Profile, XML Encryption, XML Signature, and SAML. The advanced standards include WS-Trust, WS-SecurityPolicy, WS-Federation, XML Key Management, and WS-SecureConversation.

XML Signature and XML Encryption

- Messages can be digitally signed using XML Signature.
 - The XML Signature specification defines XML syntax and processing rules for creating and representing digital signatures.
 - It can be used to sign an entire XML document or selected parts (elements) within the document.
- XML Encryption defines a process for encrypting and decrypting information.
 - It also defines an XML syntax used to represent the encrypted content and information that enables an intended recipient to decrypt it.
 - XML encryption supports the encryption of entire XML documents or individual elements within a document.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

XML Encryption is used to provide confidentiality protection for portions of a SOAP envelope, and XML Signature is used to provide integrity protection for the same. XML Signature and XML Encryption are fundamental technologies for securing XML and are pillars of WS-Security.

Security Assertion Markup Language (SAML)

Security Assertion Markup Language (SAML):

- Is an open framework for exchanging security information between different parties through XML documents
- Conveys information about subjects (human users or entities) with the following types of “assertions”:
 - Authentication
 - Authorization decision
 - Attribute



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

With WS-Security, you can use any security token you want. SAML is one of the most interesting security tokens. It is an open framework for exchanging security information between different security domains (parties), especially authentication details such as user login information, user attributes, and authorization information in the form of policies. It enables different security service systems to interoperate.

SAML is a protocol that does not define any new approaches to authentication or authorization. It simply generates appropriate tokens and assertions after authentication occurs.

SAML contains one or more “assertions,” each of which conveys information about subjects (human users or any entities). Each assertion may contain multiple statements about authentication, authorization, and additional attributes.

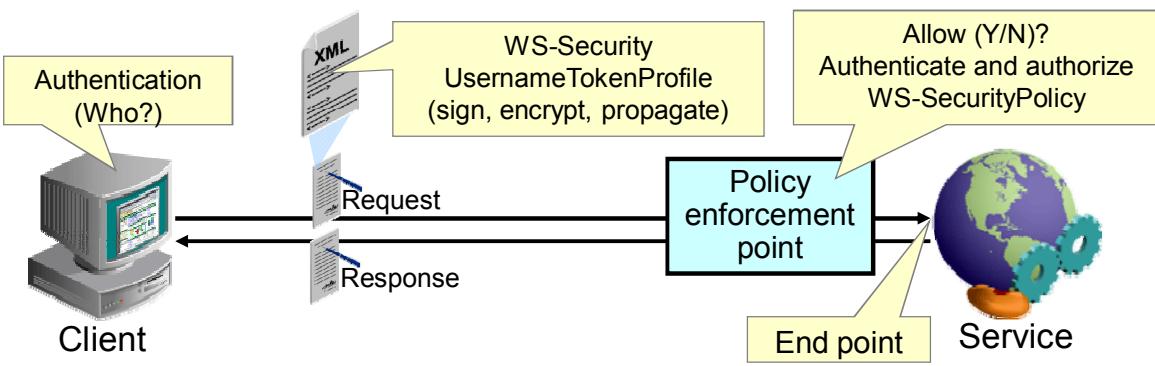
SAML supports the following types of assertions:

- **Authentication:** Conveys information that a certain subject was authenticated at a certain instant
- **Authorization decision:** Conveys information about access privileges for a user to certain resources
- **Attribute:** Conveys information about subject attributes

The SAML specification defines a standard, XML-based approach for passing security tokens defining authentication and authorization rights. SAML was originally targeted more toward distributed authentication and single sign-on (SSO), but those concepts are now also central to web services security.

WS-Security: Overview

- The goal of Web Services Security (WS-Security) is to provide comprehensive end-to-end security at message level.
- WS-Security specifies rules to ensure:
 - Authentication, using security tokens
 - Data confidentiality, using XML Encryption specification
 - Data integrity, using XML Signature specification



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE

Web Services Security (WS-Security) is the most important Web Services standard to achieve message security.

WS-Security is a collection of protocols that specify how different levels of security can be enforced on messaging in SOAP-based web services. It is meant to provide comprehensive end-to-end message content security, and not just transport-level security. Security matters are not delegated to the transport level, but rather are handled directly through an appropriate security API.

WS-Security defines how security tokens are contained in SOAP messages, how XML Security specifications are used to encrypt and sign these tokens, and how to sign and encrypt other parts of a SOAP message.

- **Data confidentiality:** WS-Security uses the XML Encryption specification to encrypt portions of SOAP messages. Any portions of SOAP messages, including headers and body blocks, may be encrypted.
- **Data integrity:** Is 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.

- **Authentication:** Can be done using security tokens. WS-Security is open to various security token models such as the following:
 - X.509 certificates
 - Kerberos tickets
 - Credentials such as user ID–password combinations
 - SAML-Assertion

WS-Security defines how to attach XML Signature and XML Encryption headers to SOAP messages, as well as how to associate security tokens with a message.

Using the WS-* security standards, much of the configuration is declarative, removing most requirements for adding security logic to the code. The key benefit of a declarative approach is the ability to change things at post-deployment time (that is, no code changes).

WS-Security and SAML

- WS-Security and SAML work together:
 - WS-Security defines how you insert the information into a SOAP envelope.
 - SAML defines what the security information is.
- SAML Token Profile 1.1 specifies how SAML assertions can be used for WS-Security:
 - WS-Security enables SAML assertions to be placed inside a SOAP header.
 - SAML Token Profile 1.1 is supported only through WS-SecurityPolicy.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

WS-Security enables SAML assertions to be placed inside a SOAP header, laying the foundations of SSO for web services.

SAML has already adopted WS-Security as the appropriate method for “binding” SAML assertions to SOAP messages.

WS-Security is the messaging language; SAML is the security language.

SAML Token Profile 1.1 is part of the core set of WS-Security standards. It specifies how SAML assertions can be used for Web Services Security. 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).

Best Practices for Applying Security

- Externalize security: Separate security from service implementation.
- Apply layered security: Implement security on various levels.
- Use security standards: Achieve reusability and interoperability.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- **Externalize security:** It is a good design principle to externalize security from service implementation. There may be different security requirements for the same service based on the type of service client, location, and network used to invoke the service. Implementing security as an integral part of a service can result in inflexible and under- or over-secured services. So security-related information such as the use of SSL/TLS and WS-Security is normally not advertised in a web service's WSDL document. The gateway and security policy are the strategies to promote the separation of concerns. These topics are discussed in later lessons.
- **Apply layered security:** Use a layered approach on various levels to secure an organization against a wide variety of security threats. Layered security involves measures on multiple levels (for example, logical access control, network security, and so on). Trying to implement security on one level or with one measure is not enough.
- **Use security standards:** Use security standards such as SSL/TLS, SAML, X.509, and WS-Security to achieve interoperability. The use of standards results in secured services being more reusable by consumers that may have heterogeneous infrastructures.

Quiz

Web Services Security (WS-Security) is a standard to achieve security at transport level.

- a. True
- b. False



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

In this lesson, you should have learned how to describe:

- Web services security challenges
- Key web services security concepts
- XML and web services security standards



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



Getting Started with Oracle Enterprise Gateway 11g

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the role Oracle Enterprise Gateway plays in a multilayered security deployment for web services
- Describe the capabilities of Oracle Enterprise Gateway
- Describe the Oracle Enterprise Gateway components
- Become familiar with the OEG user interface
- Configure an Oracle Enterprise Gateway instance



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

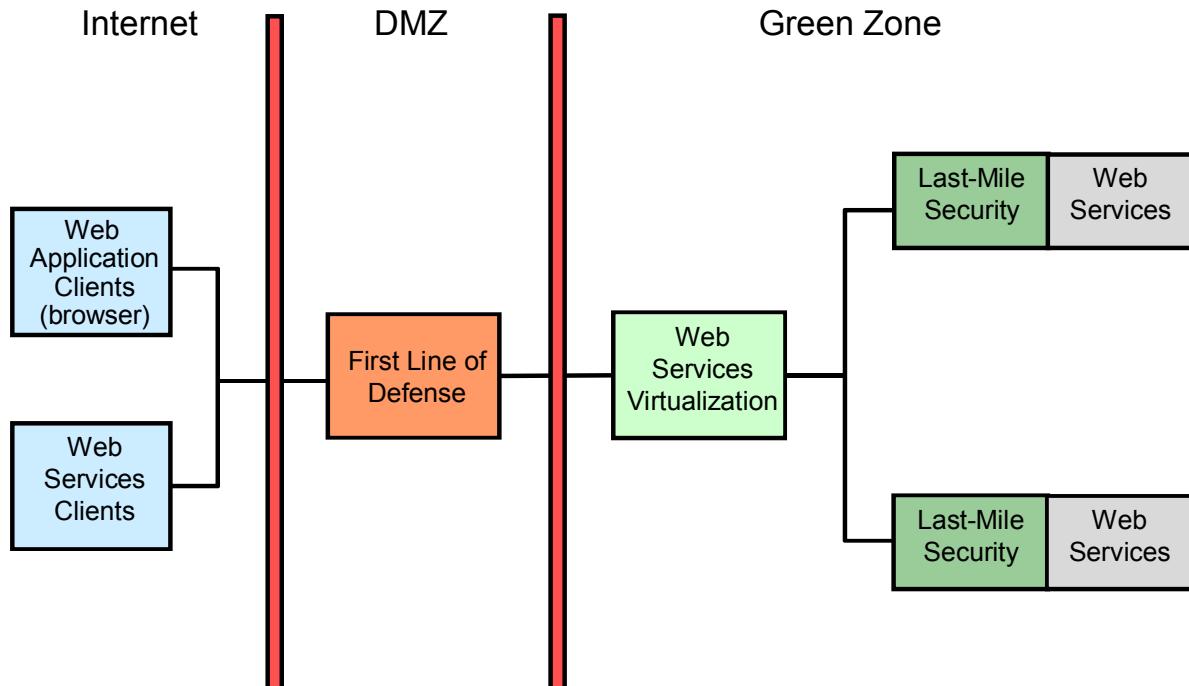
Roadmap

- OEG in a multilayered security deployment for web services
- OEG architecture and components
- Exploring the OEG user interface
- Configuring an Enterprise Gateway instance



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Problem Statement



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

SOA infrastructures are mainly deployed by using XML web services:

- XML, transformations, crypto, SSL, web service security, and processing are all CPU intensive.
- Clients may not support standards and technologies that are leveraged by back-end web services.
- There are many types of client technologies: browsers, mobile devices, application clients, and so on.
- No client should be able to consume all back-end CPU cycles. There is a need to charge for usage.

Web services in the DMZ/network perimeter layer are highly exposed:

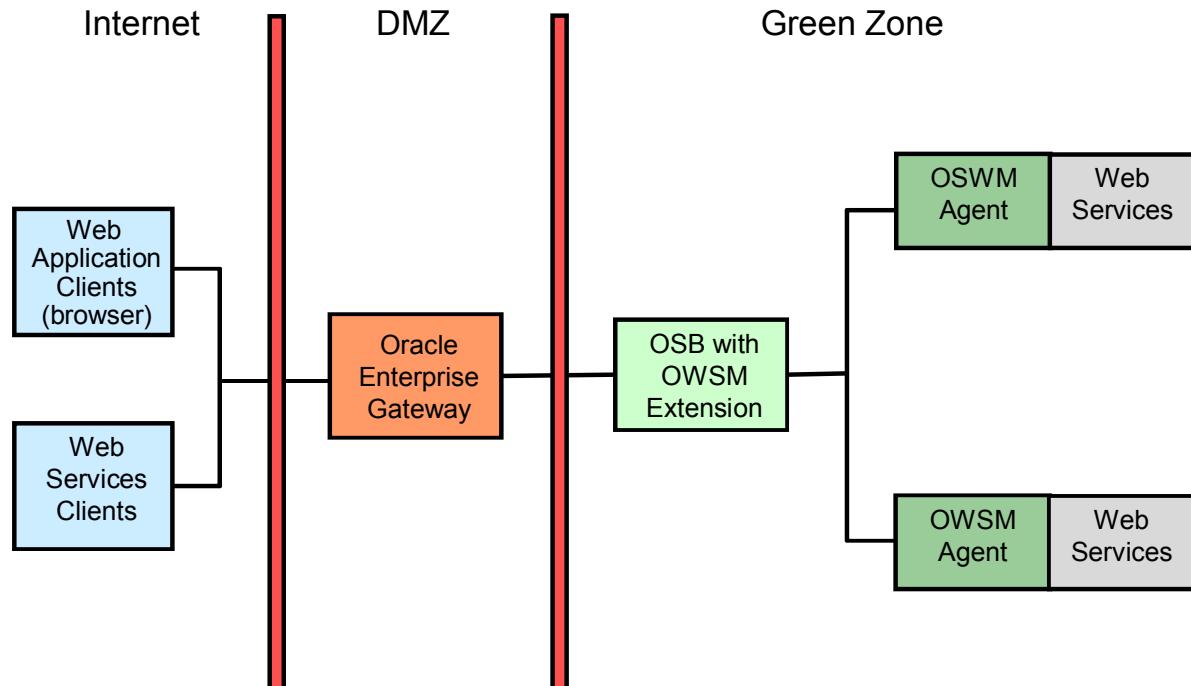
- They must be protected from XML threats, viruses, and DoS attacks. You must be able to ensure confidentiality and non-repudiation.
- You must be able to control who can access the service, and under what conditions.
- You must be able to control what data is leaving the network, and how it leaves.

Web services can be implemented by using different approaches and technologies that need to be secured at different stages of the request-response cycle between clients (users or applications) and service providers (companies or divisions within a company exposing web services).

Several security layers are defined between clients and web service providers:

- **Demilitarized zone (DMZ):** Also known as “perimeter security” or “first line of defense”
- **Green zone:** Located behind the inner firewall of the DMZ; in some cases, it may include several security sublayers that are designed to further filter access to web services
- **Last-mile security:** Provided by agents that are co-located with the web services or applications to be protected

Oracle Solution



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In a typical deployment scenario, Oracle Enterprise Gateway components are deployed in the demilitarized zone (DMZ). The connection between the client and the Enterprise Gateway is protected by a perimeter firewall, and the connection between the Enterprise Gateway and the web service is protected by a Network Address Translation (NAT) firewall.

Web service virtualization is generally provided by a service bus (in this case, Oracle Service Bus). The last-mile security is ensured by agents directly injected in the web service application so that there is no unprotected space between the first line of defense and the requested web service (unprotected space can lead to “man-in-the-middle” attacks where an attacker inserts itself in the communication channel between client and web service provider, masquerading as the actual web service requester).

Oracle Enterprise Gateway can be deployed as a stand-alone or an integral component of a strategic enterprise SOA infrastructure, interfacing with enterprise service bus, enterprise management, and identity management platforms.

Note: The diagram illustrates a deployment architecture that provides the highest security to web services. For some use cases, customers simply use OEG and have web services deployed directly on top of an application server, such as WebLogic Server.

Basics of OEG

- Provides XML-specific firewall to address the security challenges of XML and web services
- Protects back-end web services through service registration
- Uses policies to provide security, routing, and transformation functionality
- Serves static content and servlets



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

At its core, OEG is an XML firewall. The XML firewall is designed to examine and evaluate the XML content of incoming traffic and, based on that evaluation, perform one or more of the following security actions as appropriate:

- Routing the message to a designated endpoint
- Transforming the message based on its content
- Validating a signature
- Decrypting a field
- Blocking access to certain operations

When you register a web service and deploy it to the Enterprise Gateway, the Enterprise Gateway virtualizes the web service. Instead of connecting to the web service directly, clients connect through Enterprise Gateway. The Enterprise Gateway can then apply policies to messages sent to the destination web service (for example, to enable security, monitoring, and acceleration). Policies contain specific assertions about operational attributes, such as authentication and authorization, encryption and signatures, and routing and transformations.

The Enterprise Gateway provides a web server and servlet application server that can be used to host static content (for example, documentation for your project) or to host servlets providing internal services.

OEG Features

- Application-level routing
- XML conversion and validation
- XML threat blocking
- XML acceleration
- XML security
- Monitoring
- Governance



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Oracle Enterprise Gateway is a software solution that provides:

- Application-level routing (based on source, target, sender identity, and XML message type)
- XML conversion, validation, and threat scanning
- XML acceleration
- Security (selective encryption and signature of XML messages, decryption, and signature validation)
- Monitoring (response time, logging, and alerting)
- Governance (service access and usage)

Quiz

Which of the following features is not provided by OEG 11g?

- a. XML firewalling
- b. XML accelerating
- c. Load balancing
- d. Application-level routing



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

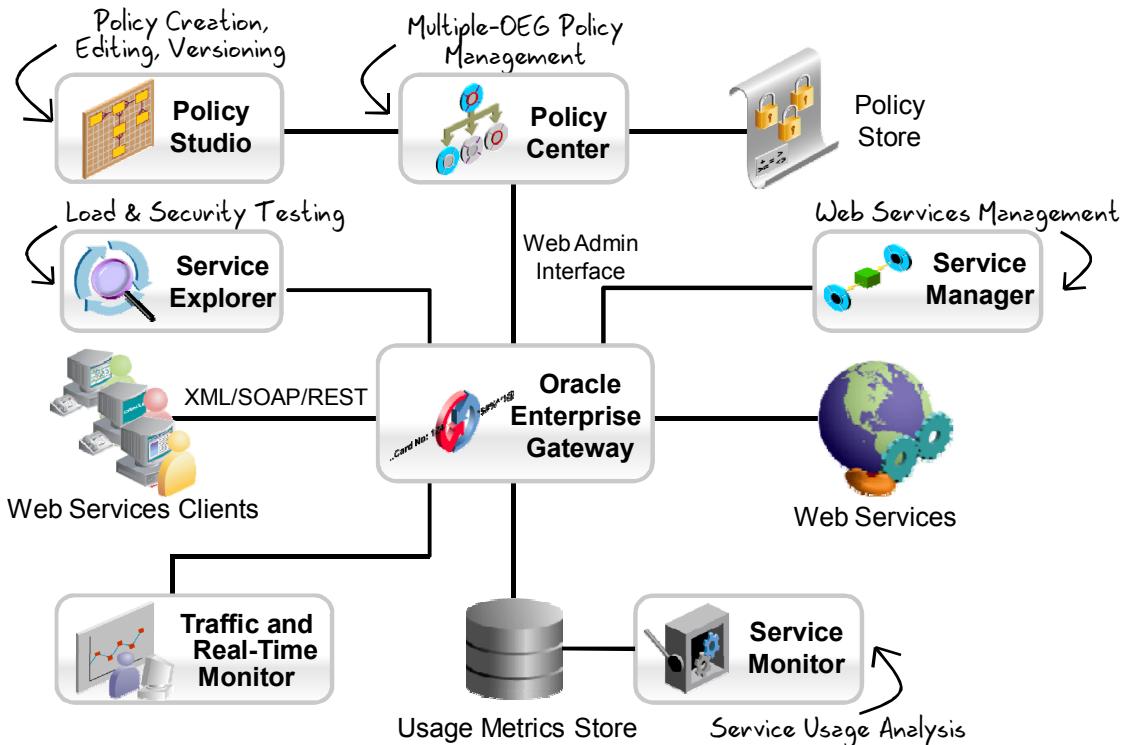
Roadmap

- OEG in a multilayered security deployment for web services
- OEG architecture and components
- Exploring the OEG user interface
- Configuring an Enterprise Gateway instance



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

OEG Architecture and Components



ORACLE

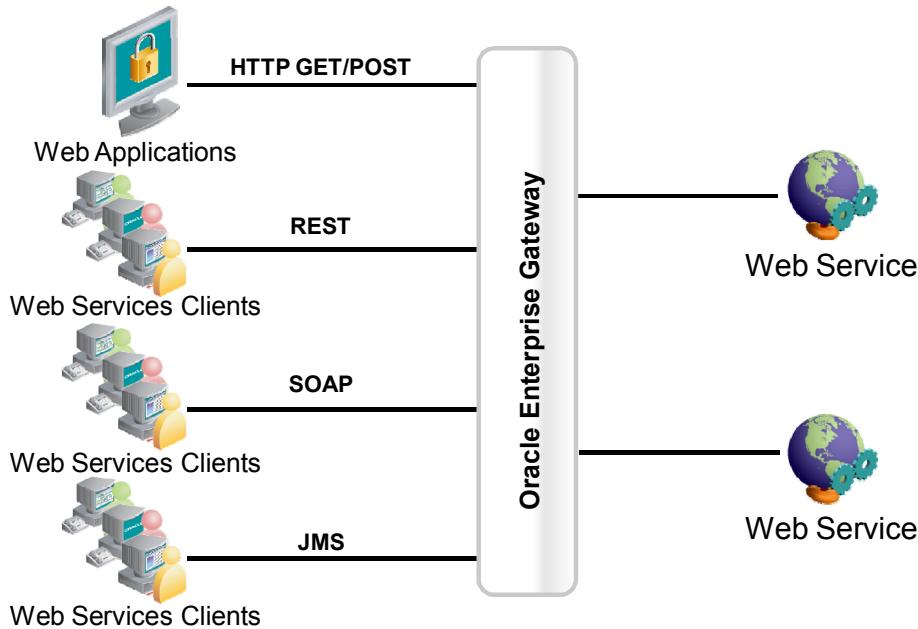
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Oracle Enterprise Gateway comprises several components, as illustrated in the slide. At the core is the Enterprise Gateway, which provides governance, acceleration, integration, and security for web services. The Enterprise Gateway applies policies to incoming messages by running message filters on requests. The rest of the components are designed for testing, managing, and monitoring purposes.

- **Policy Studio**: An authoring tool for creating and editing all policies
- **Policy Center**: A configuration hub for centralized policy definition and deployment. When connected to a Policy Center server, you can deploy configurations to different Enterprise Gateway servers running in the network.
- **Service Explorer**: A web services test client to generate test messages that are sent to the Enterprise Gateway and back to Service Explorer
- **Service Manager**: A web-based system administration tool that simplifies Enterprise Gateway management tasks. It provides quick and easy access to enable you to manage your services and policies. For example, you can register web services and assign policies to them.

- **Real-Time Monitoring Console:** Provides web-based real-time monitoring of HTTP and HTTPS traffic processed by the Enterprise Gateway. This web-based console enables administrators to detect malicious activity in real time and take precautionary actions if they feel a service is under attack.
- **Service Monitor:** A separately installed component that generates reports and charts based on the usage metrics of all the Enterprise Gateways in your network. Service Monitor provides integration with databases such as MySQL Server, MS SQL Server, and Oracle database.

Enterprise Gateway Component



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Oracle Enterprise Gateway is available as a single executable for the Microsoft Windows, Linux, and Oracle Solaris platforms. The Enterprise Gateway performs application networking by routing traffic based on content and sender, and by performing XML content screening. The Enterprise Gateway applies policies to incoming messages by running message filters on requests.

The diagram shows that the Enterprise Gateway supports a wide range of transports and protocols, such as HTTP, REST, SOAP, JMS, TIBCO, FTP, SMTP, and POP.

OEG Policy Studio

- Policy Studio is a policy management tool for the following tasks:
 - Creating and editing policies
 - Configuring Enterprise Gateway settings
- It supports the following features:
 - Drag-and-drop policy configuration
 - Specifying different policy configuration versions
 - Pushing policies to one gateway or (with Policy Center) to multiple gateways



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Policy Studio is a policy management tool that enables an administrator to easily configure policies and Enterprise Gateway settings to control and protect all deployed web services.

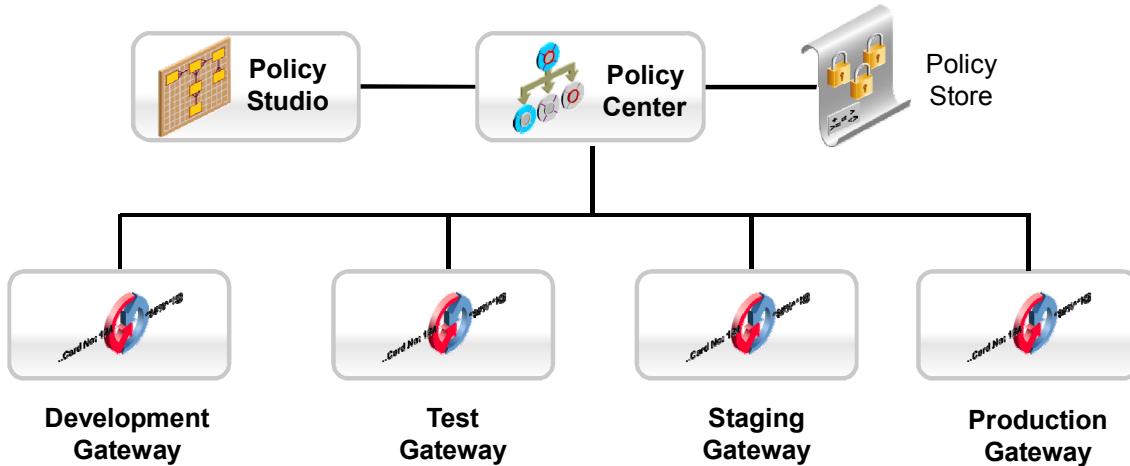
For example, Policy Studio enables you to do the following:

- Create and assign policies
- Configure the full range of Enterprise Gateway configuration settings
- Manage your Enterprise Gateway deployments

To enable remote administration, Policy Studio is typically installed on a machine that is separate from the Enterprise Gateway.

OEG Policy Center

Policy Center enables centralized policy management across a distributed network of Enterprise Gateways.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Policy Studio enables policies to be created and versioned. When connected to a Policy Center server, you can then push out this configuration to Enterprise Gateways that have been deployed throughout the organization. Policy Studio and Policy Center address the need for centralized policy management across a distributed network of Enterprise Gateways. These tools enable enterprises to control a distributed network of Enterprise Gateways from a single location.

With Policy Center, you can:

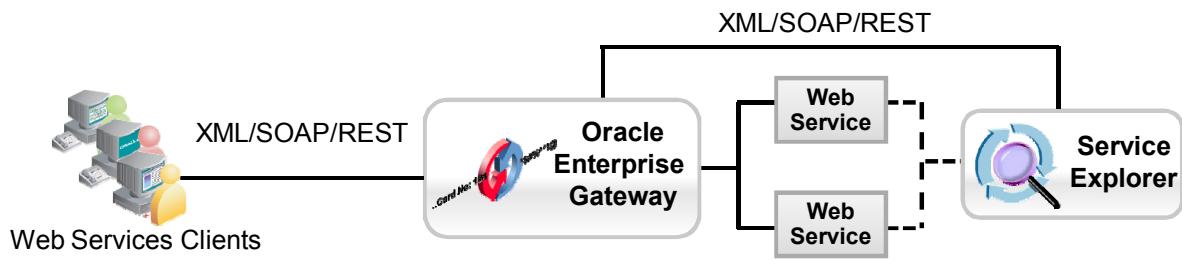
- Configure multiple gateways together
- Store policies in file systems or databases
- Manage policy life cycles
- Provision policies to new Enterprise Gateways
- Monitor the audit trail of policy changes

One of the most powerful uses of this centralized management capability is in transitioning from a development environment to a production environment. The slide depicts using a Policy Center server so that policies can be easily migrated between development, testing, staging, and production Enterprise Gateways.

OEG Service Explorer

A web services test client that has the following capabilities:

- Testing and stress-testing web services before deployment
- Testing XML, REST, and SOAP messages
- Adding SOAP attachments easily
- Simple graphical keystore
- Configuring XML encryption and decryption
- Inserting/removing security tokens



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

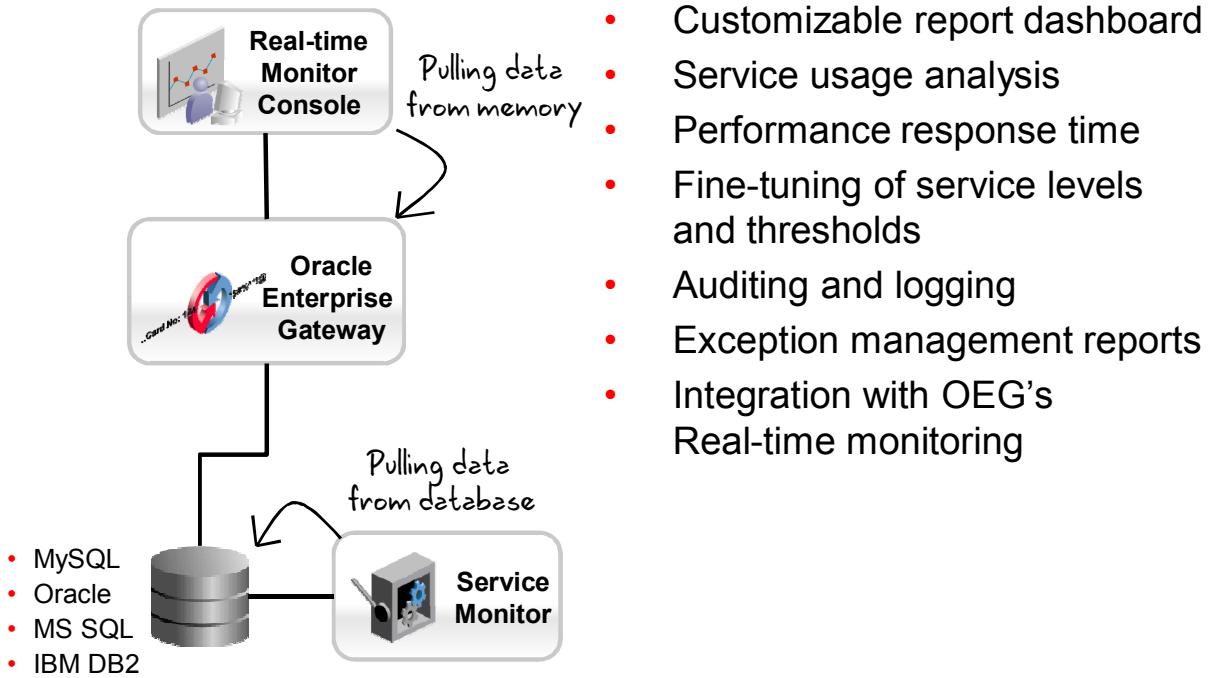
OEG Service Explorer is a web services test client that can automatically generate request messages from the service interfaces defined in WSDL files. Using Service Explorer, developers can test the performance, scalability, and security of web services. Service Explorer has the following capabilities:

- **Testing and stress-testing web services before deployment:** Service Explorer allows you to create test suites consisting of several test cases for each service defined in the WSDL. Test cases can then be run in parallel in order to stress-test a target web service.
- **Testing XML, REST, and SOAP:** Service Explorer supports SOAP-based invocations. In addition, web services that are called only directly by browsers (for example, REST services) can also be tested using Service Explorer.
- **Adding SOAP attachments easily:** Service Explorer can be used to attach SOAP attachments to SOAP messages simply and easily.
- **Simple graphical keystore:** Service Explorer turns key management into a point-and-click task by using a graphical keystore. Users can load multiple keys and certificates without using command-line tools. With the keystore, it is possible to encrypt XML for multiple recipients in one easy step.

- **Configuring XML encryption and decryption:** Service Explorer can create signed and encrypted XML messages without any requirement for coding.
- **Inserting and removing security tokens:** Service Explorer enables the rapid insertion and removal of tokens to facilitate speedy testing. It supports WS-Security and SAML tokens.

Note: Service Explorer also provides a command-line option for stress testing.

OEG Service Monitor



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

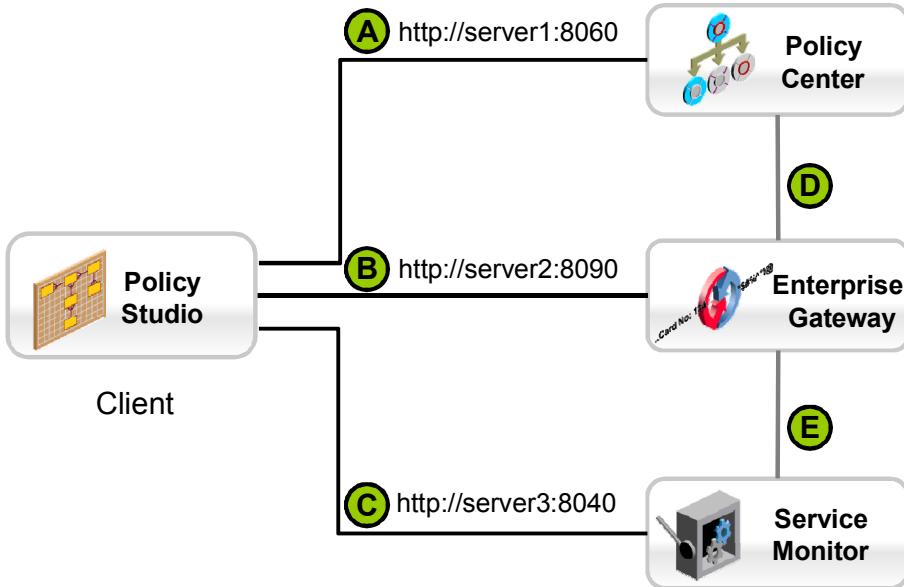
Service Monitor is a separately installed component that generates reports and charts based on the usage metrics of all the Enterprise Gateways in your network. For example, you can generate and store reports that monitor which authenticated clients are calling which web services.

Service Monitor reads message metrics from a database and displays this information in a visual format to administrators.

The Enterprise Gateway stores and maintains the monitoring and transaction data that is read by Service Monitor in a JDBC-compliant database. Service Monitor provides setup scripts for databases: MySQL, Oracle, MS SQL, and IBM DB2.

Service Monitor also provides a web-based console for monitoring real-time statistics.

Connections Between Components



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The diagram shows the default connections of a typical deployment of Policy Studio, Enterprise Gateway, Policy Center, and Service Monitor. For your practices, all components are installed on one machine.

The connections can be described as follows:

- A:** Policy Studio connects to Policy Center over HTTP on port 8060.
- B:** Policy Studio connects to the gateway over HTTP on port 8090.
- C:** Policy Studio connects to the Service Monitor server over HTTP on port 8040.
- D:** Policy Center connects to the gateway management services interface over HTTP on port 8090 to push/pull configurations to and from the gateway.
- E:** Service Monitor connects to the gateway over HTTP on port 8090.

Quiz

Policy Studio enables you to do the following (select all that apply):

- a. Creating, editing, and versioning policies
- b. Deploying configurations to multiple Enterprise Gateway server instances
- c. Testing XML, REST, and SOAP messages
- d. Configuring Enterprise Gateway settings
- e. Monitoring the Enterprise Gateway



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, d

Roadmap

- OEG in a multilayered security deployment for web services
- OEG architecture and components
- Exploring the OEG user interface
- Configuring an Enterprise Gateway instance



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Starting the OEG Components

- Enterprise Gateway (a daemon process)
`<oeginstall_dir>/enterprisegateway posix/bin/
enterprisegateway`
- Policy Studio (a Java application)
`<oeginstall_dir>/oegpolicystudio/policystudio`
- Policy Center (a daemon process)
`<oeginstall_dir>/oegpolicycenter posix/bin/
oegpolicycenter`
- Service Explorer (a Java application)
`<oeginstall_dir>/oegserviceexplorer/serviceexplorer`
- Service Monitor (a daemon process)
`<oeginstall_dir>/oegservicemonitor posix/bin/
oegservicemonitor`

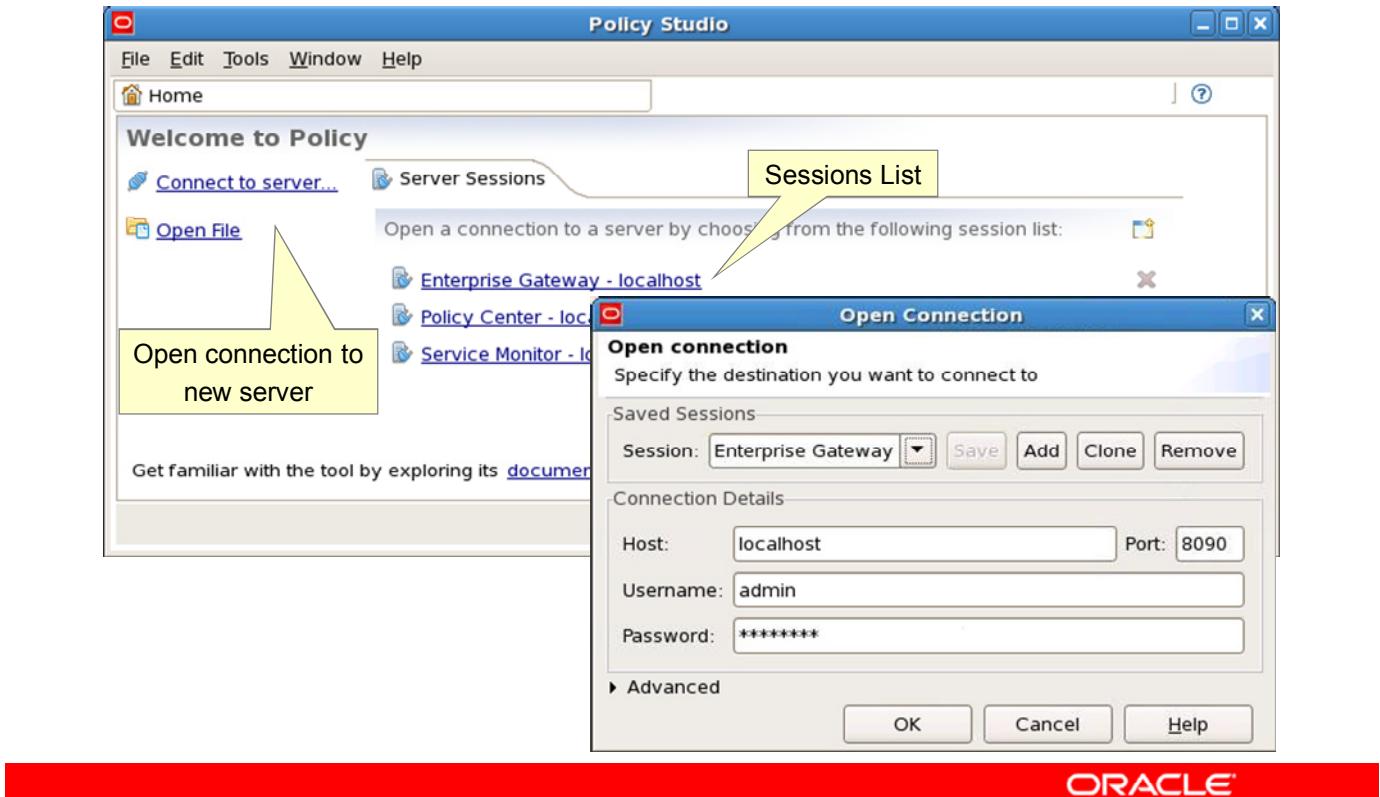


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The OEG installation is simple and straightforward. You just need to extract the compressed file of each component to a suitable location. After installation, use the commands listed in the slide to start each component.

Note: Service Monitor reads message metrics from a database and generates reports and charts, so a JDBC-compliant database needs to be set up before you start Service Monitor.

Connecting Policy Studio to Gateway



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can use Policy Studio to manage Enterprise Gateway, Policy Center, and Service Monitor servers. Before making the connection, ensure that the server process that you want to connect to has been started.

When Policy Studio starts up, click a link to a server to display the Open Connection dialog box. You can use this dialog box to specify Connection Details or to specify Saved Sessions.

In the Open Connection dialog box, the Connection Details section allows you to specify the following settings:

- **Host:** Specify the host to connect to in this field. The default is `localhost`.
- **Port:** Specify the port to connect on in this field. In this example, the default Enterprise Gateway port is `8090`.
- **User Name:** The deployment service is protected by HTTP Basic authentication. You must provide a username and password so that Policy Studio can authenticate to the server. By default, the server User Store contains an `admin` user with a `changeme` password, which is used in this case. You can change this user's details by using the User Store interface.
- **Password:** Specify the password for the user. The password for the default admin user is `changeme`.

If you want to connect to the server by using a non-default URL, click Advanced and enter the URL. The default Enterprise Gateway server URL is:

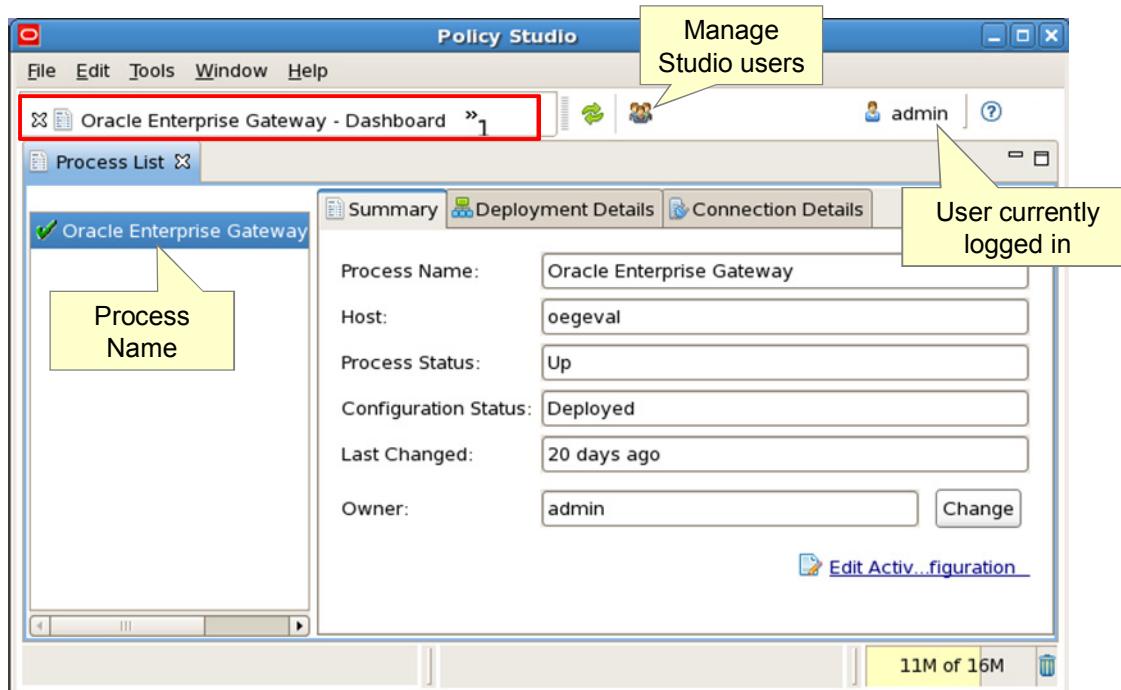
<http://localhost:8090/configuration/deployments/DeploymentService>

Alternatively, you can connect to a server configuration file by clicking the Open File button.

By default, the configuration file `configs.xml` is located in the

`<oeginstall_dir>/enterprisegateway/conf/fed` directory.

Oracle Enterprise Gateway Dashboard



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When the connection to the server has been made, the Oracle Enterprise Gateway Dashboard tab is displayed. The Dashboard tab enables you to manage the configuration of currently running Enterprise Gateway instances, which are called *processes*. You can use Policy Studio to configure and deploy Enterprise Gateway processes. Managed processes can include Enterprise Gateway, Policy Center, and Service Monitor processes.

All available processes are displayed in the left pane, and more detailed information about a deployed process is in the right pane, which includes summary, deployed configuration, and connection details. You can perform tasks such as changing a process owner, updating connection details, and applying tags on the deployed process.

Policy Studio Users Versus Enterprise Gateway Users

- Policy Studio Users: Provide access to the configuration and process management features in Policy Studio
- Enterprise Gateway Users: Provide access to the messages and services protected by the Enterprise Gateway



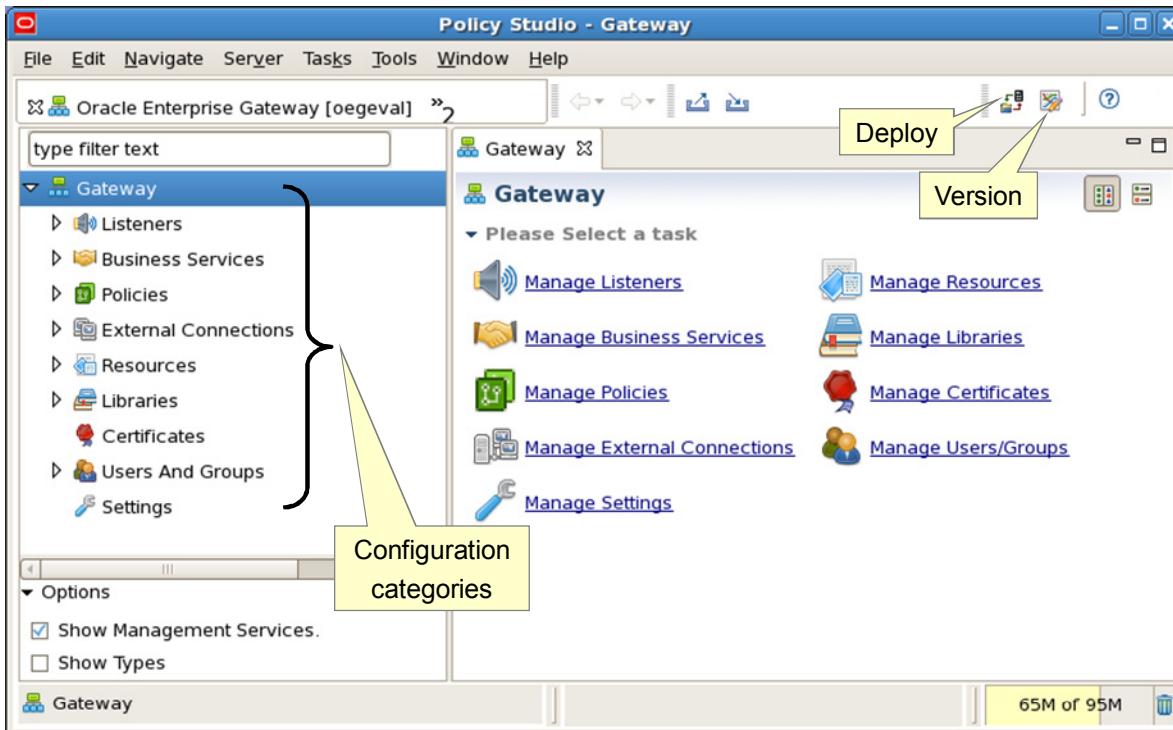
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Policy Studio has its own user store. This means that when logging in to Policy Studio to access a particular process, you must enter the credentials of a user stored in its user store to enable Policy Studio to connect to the process.

Policy Studio Users are responsible for managing server processes, configuration profiles, and—in the case of Super Users—managing other Policy Studio Users. You can manage Policy Studio Users by clicking the Users button at the top left of the Oracle Enterprise Gateway Dashboard.

Enterprise Gateway Users specify the user identity in the User Store. This includes details such as the username, password, and X.509 certificate. An Enterprise Gateway User must be a member of at least one User Group. To view all existing Users, select the Users and Groups > Users node in the Policy Studio tree.

Editing Active Server Configuration



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

For convenience, Policy Studio contains various global configuration options. For example, it includes libraries of users, X.509 certificates, and schemas that can be added globally and then referenced in filters and policies. This avoids the need to reconfigure details over and over again (for example, each time a schema or certificate is used).

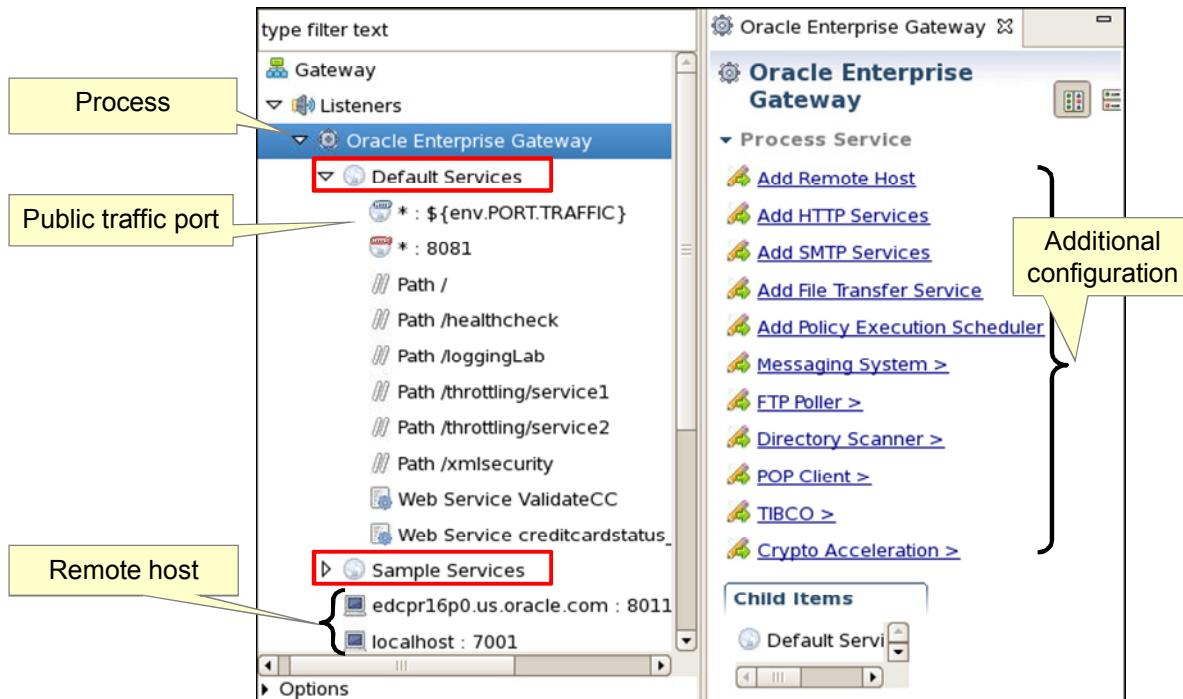
The various global configuration options are categorized as follows:

- **Listeners:** Store the definition of HTTP Interface with the address and port that the Enterprise Gateway process listens on
- **Business Services:** Store information about web services whose definitions have been imported using Policy Studio or Service Manager
- **Policies:** Store policies. You can use a Policy Container to group together a number of similar policies or to act as an umbrella around several policies that relate to a particular policy.
- **External Connections:** Store global configuration details about third-party servers (for example, LDAP directories, databases, Identity Management Servers, and so on) that the Enterprise Gateway connects to in order to process requests
- **Resources:** Store scripts and stylesheets that the Enterprise Gateway can use to interact with and transform incoming request messages, and XML Schemas that can be used globally by Schema Validation filters

- **Libraries:** Contain Black list, White list, caches, and alerts
- **Certificates:** Contain all the certificates that are considered to be trusted by the Enterprise Gateway. Certificates can be imported into or created by the Certificate Store.
- **Users And Groups:** Contain the configuration data for managing Enterprise Gateway user information
- **Settings:** Configure the underlying settings for the Enterprise Gateway

When editing an active server configuration, you can deploy updates by clicking the Deploy button in the toolbar. You can version deployed updates by clicking the toolbar's Version button.

Listeners



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

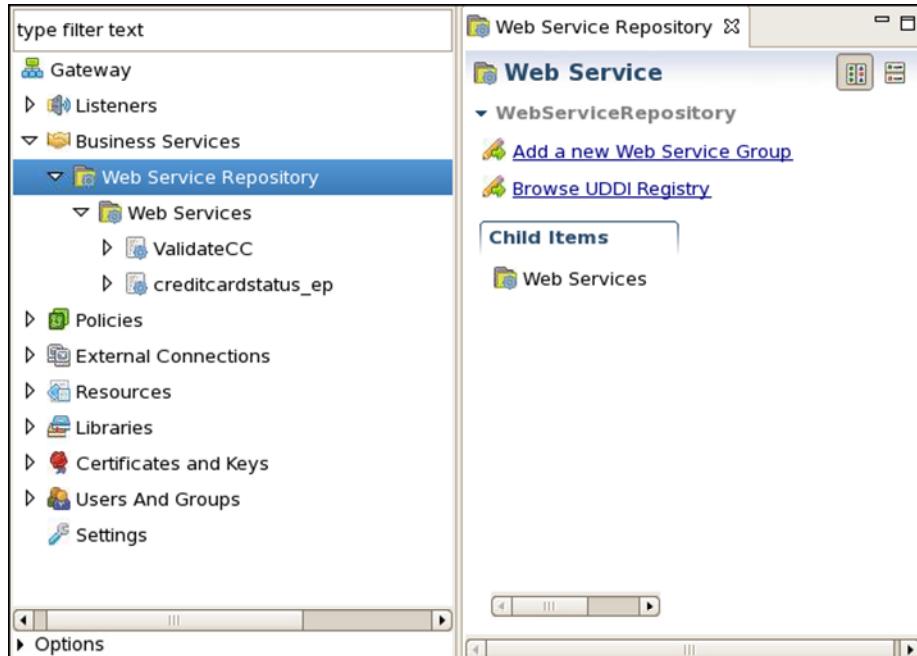
Under the **Listeners** node, you can configure the process. A process represents a single running instance of the Enterprise Gateway.

By default, there are three interfaces: one for public traffic (under the Default Services node), one for a sample service (under the Sample Services node), and one for listening to and serving configuration data (under the Management Services node). Out of the box, the public traffic interface port is 8080, the sample service interface port is 8081, and the management/configuration interface port is 8090. The configuration interface should rarely need to be updated, so the Management Services node is not visible by default. However, it is likely that you will want to add several HTTP interfaces (under the Default Services node). For example, you may want to add an HTTP interface and an SSL-enabled HTTPS interface.

Furthermore, you can also add features such as the following at the process level:

- Remote hosts to configure the way in which the Enterprise Gateway connects to a specific external server or routing destination
- SMTP interfaces to configure email relay
- Policy execution schedulers to run policies at regular time intervals
- JMS listeners to listen for JMS messages
- Packet sniffers to inspect packets at the network level for logging and monitoring
- Directory scanners to scan messages dumped to the file system

Web Service Repository

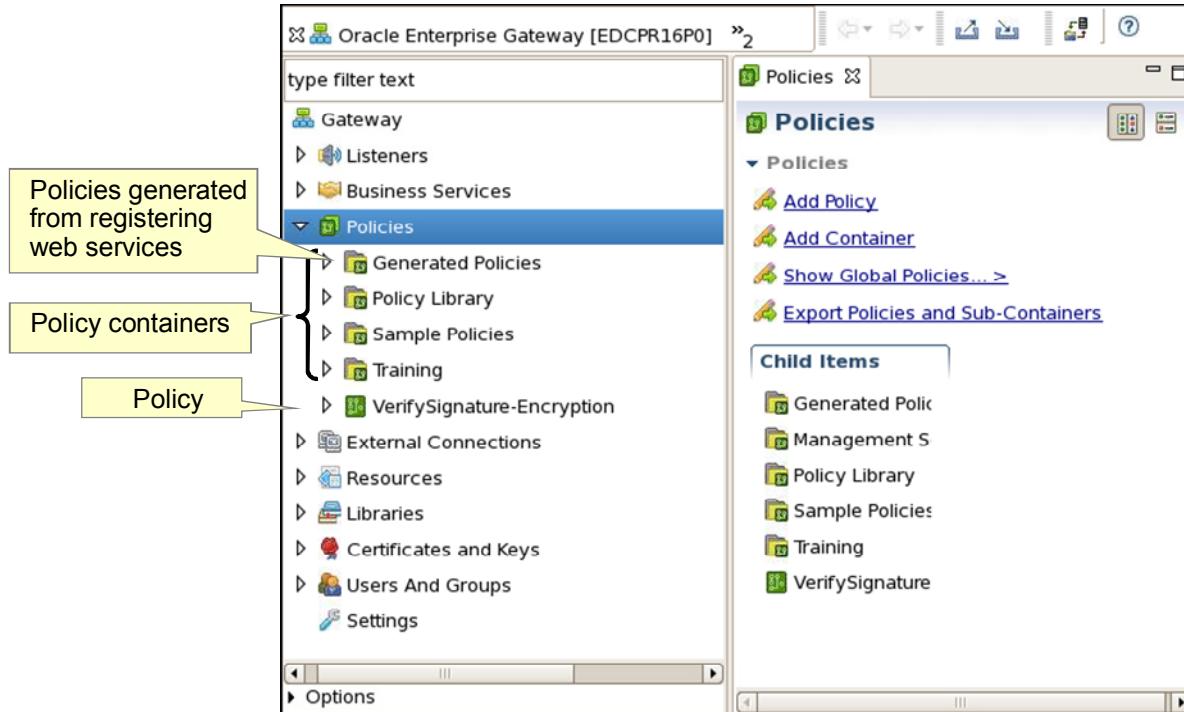


ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The **Web Services Repository** (under the Business Services node) stores information about web services whose definitions have been imported using Policy Studio or Service Manager. The WSDL files that contain these web services definitions are stored with their related XML Schemas. Clients of the web service can then query the repository for the WSDL file, which they can use to build and send messages to the web service by using the Enterprise Gateway.

Policies



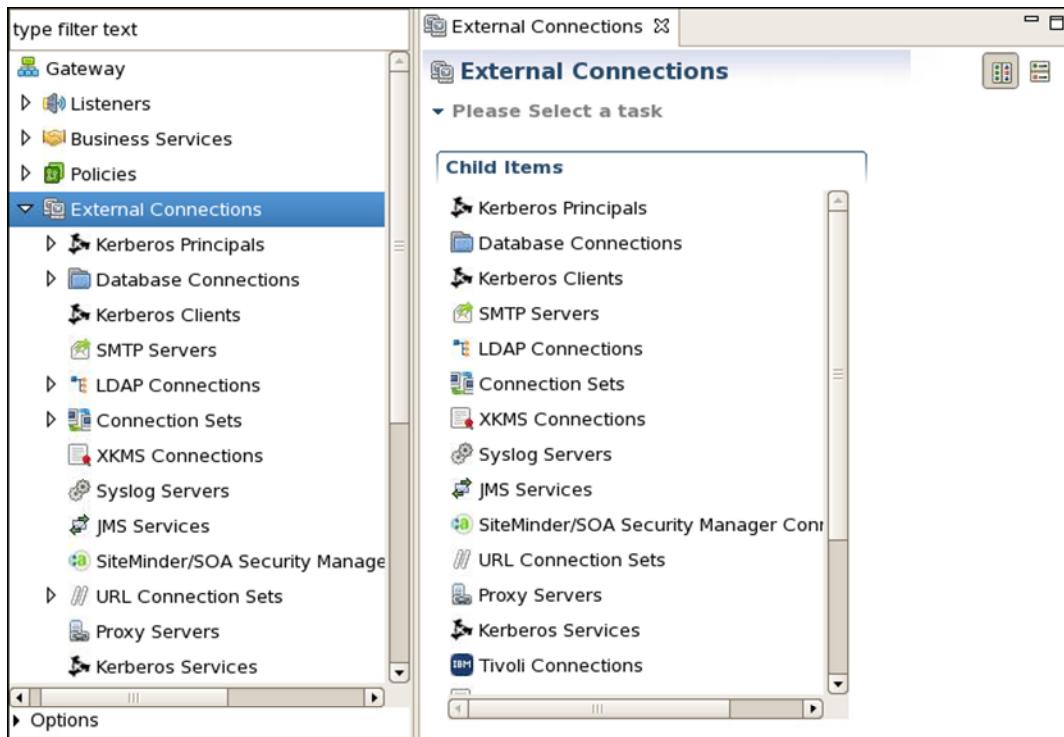
ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Policies are managed under the **Policies** node. You can create containers (and subcontainers) to organize your policies, for example, by project. A policy can be reused from another one, whether they are stored in the same container or not.

A number of useful policies that ship with the Enterprise Gateway are found in the Policy Library container. This container is prepopulated with policies to return various types of faults to the client and policies to block certain types of threatening content, among others. You can also add your own policies to this container, and create your own Policy Containers as necessary to suit your own requirements.

External Connections



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

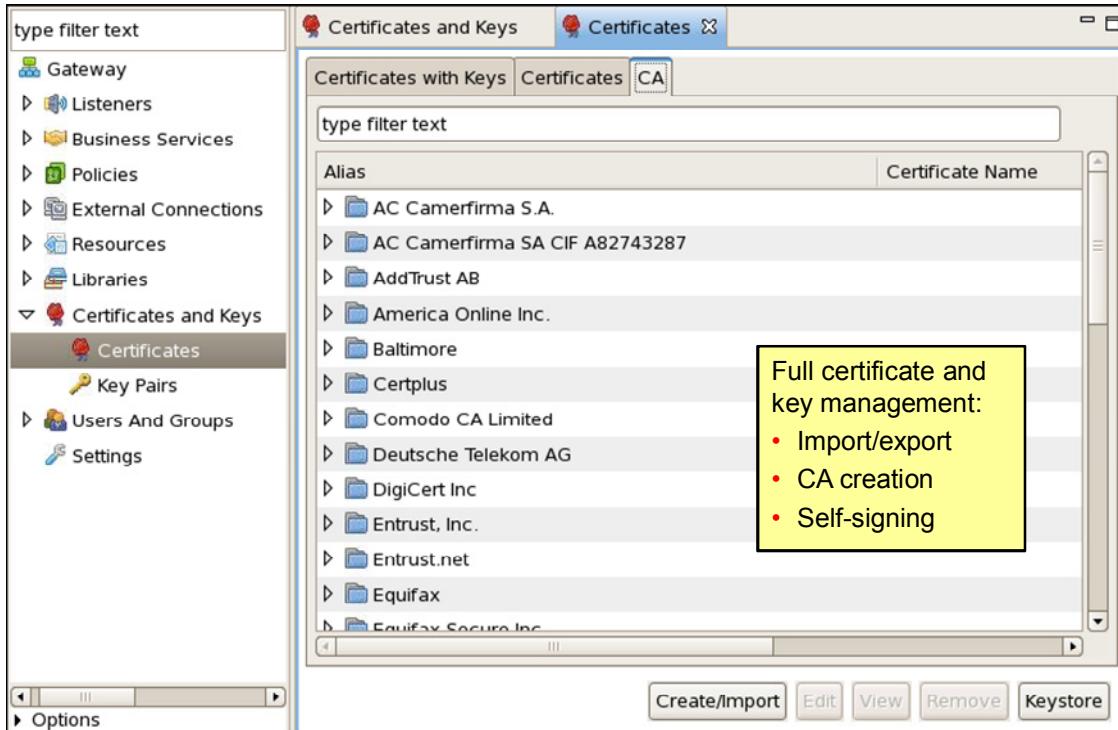
Under the **External Connections** node, you can add a connection to an external system (for example, LDAP directories, databases, Identity Management Servers, and so on) so that it can be reused across all filters and policies. For example, if you create a circuit that authenticates users against an LDAP directory and then validates an XML signature by retrieving a public key from the same LDAP directory, it makes sense to create a global external connection for that LDAP directory. You can then select the LDAP connection in both the authentication and XML signature verification filters, rather than having to reconfigure it in both filters.

For example, you can use the External Connections interface to configure global connections such as the following:

- Authentication Repository Profiles
- Database Connections
- Kerberos Services
- LDAP Connections
- OCSP Connections
- XKMS Connections

You can also use External Connections when you want to configure a group of related URLs. This is most useful in cases where you want to round-robin between a number of related URLs to ensure high availability. When the Enterprise Gateway is configured to use a URL Connection Set (instead of just a single URL), it round-robins between the URLs in the set.

Certificates and Keys



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

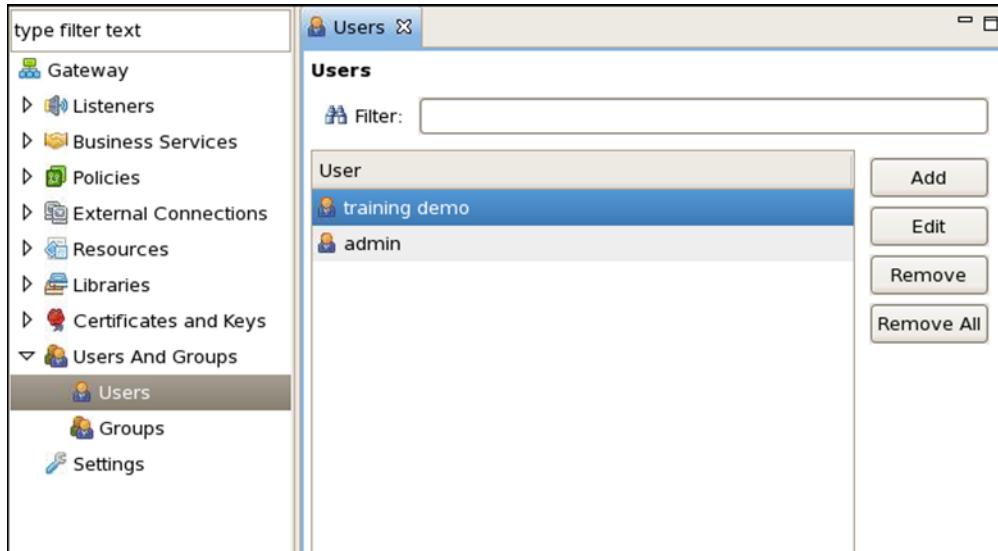
The Enterprise Gateway must be able to trust X.509 certificates to establish SSL connections with external servers, validate XML Signatures, encrypt XML segments for certain recipients, and perform other such cryptographic operations. Similarly, a private key is required to carry out certain other cryptographic operations, such as message signing and decrypting data.

The Certificate Store can be used to import, export, and create X.509 certificates, which can then be used for signing, encryption, validation, authentication, authorization, and other uses.

The Trusted Certificate Store contains all the certificates that are considered to be trusted by the Enterprise Gateway.

To assign a private key to the public key stored in a certificate, import the private key or generate one by using the provided interface.

Users and Groups



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

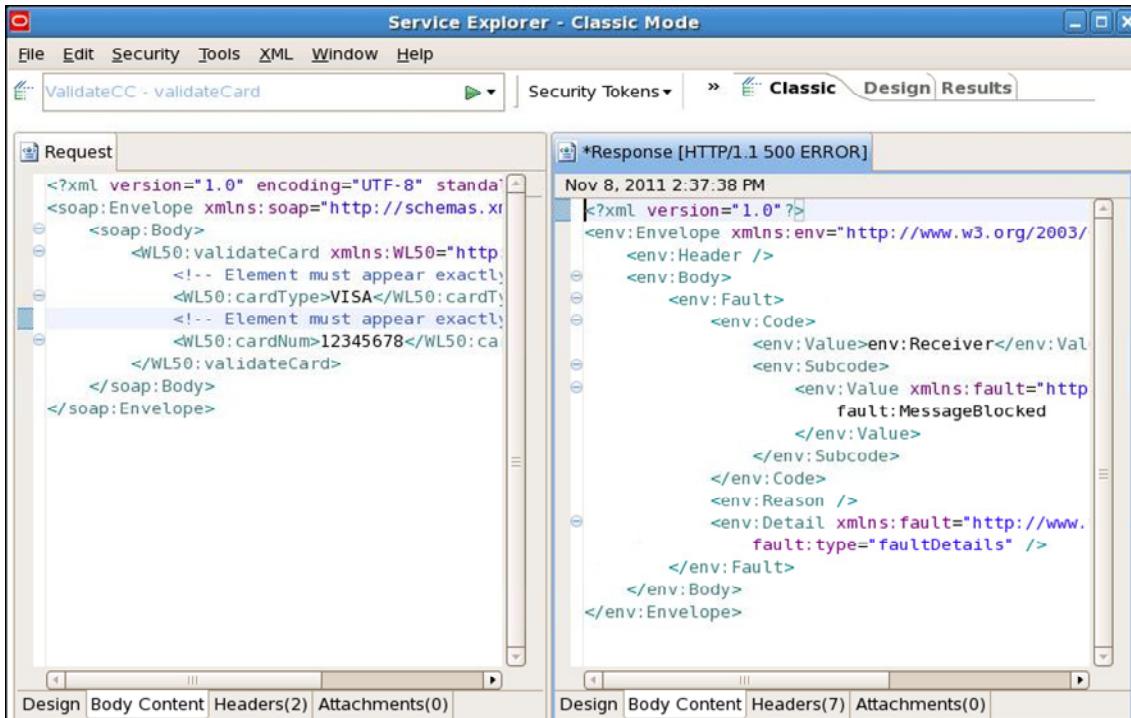
Enterprise Gateway **Users and Groups** contains the configuration data for managing Enterprise Gateway user information.

Enterprise Gateway users specify the user identity in the user store. This includes details such as the username, password, and X.509 certificate.

Enterprise Gateway user groups are containers that encapsulate one or more users. You can specify attributes at the group level, which are inherited by all group members. If a user is a member of more than one group, that user inherits attributes from all groups.

You can specify attributes at the user level and at the group level on the Attributes tab. Attributes specify user configuration data (for example, attributes used to generate SAML attribute assertions). Users can inherit attributes at the group level.

Service Explorer



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Service Explorer enables you to test how web services perform under load, how they deal with unexpected input, and what their traffic ceiling is.

Service Explorer can automatically generate request messages from the service interfaces defined in WSDL files. Application-level security can then be applied to messages by adding XML Signatures, XML Encryption segments, SAML assertions, and WS-Security tokens. Requests can then be sent to the target web service to make sure it is processing the security tokens appropriately. Traditional security technologies such as SSL and HTTP authentication are also supported.

Roadmap

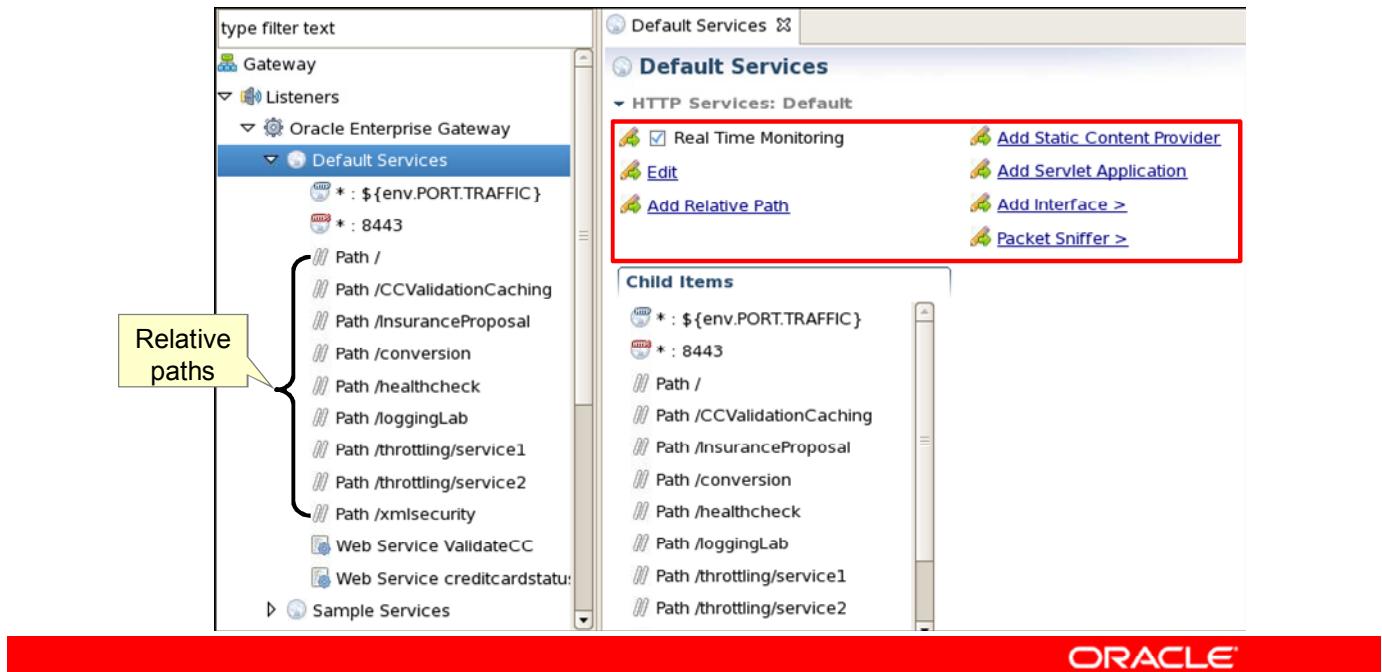
- OEG in a multilayered security deployment for web services
- OEG architecture and components
- Exploring the OEG user interface
- Configuring an Enterprise Gateway instance
 - Configuring HTTP Services
 - Adding a remote host



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Configuring HTTP Services

The Enterprise Gateway uses HTTP Services to serve traffic from various HTTP-based sources:



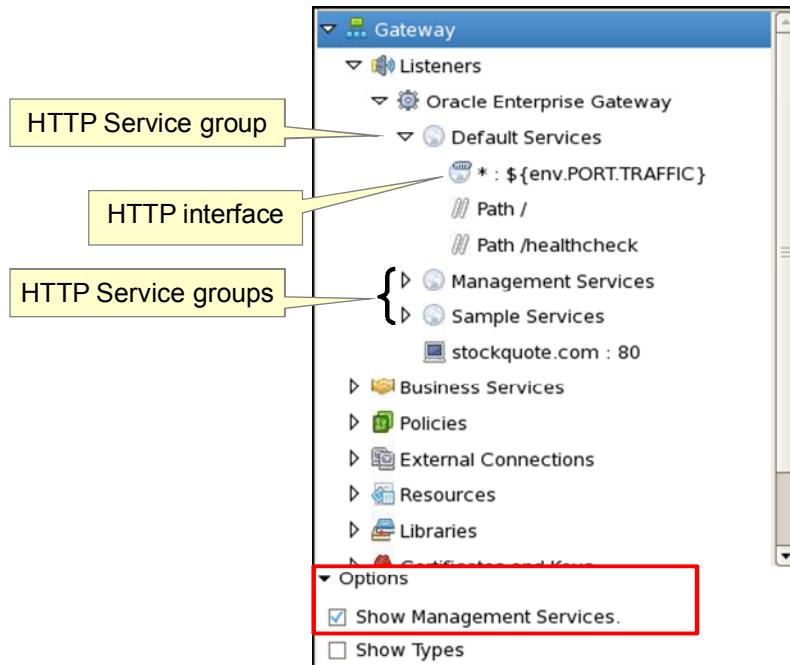
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The following list summarizes the available HTTP services:

- **HTTP interfaces:** Are used in the Enterprise Gateway to define the ports and IP addresses on which the Enterprise Gateway process listens for incoming requests. You can also add HTTPS interfaces and select the SSL certificate to use to authenticate to clients, and also select the certificates that are considered trusted for establishing SSL connections with clients.
- **Relative path:** While HTTP and HTTPS interfaces open sockets and listen for traffic, you can configure relative paths so that when a request is received on that path, the Enterprise Gateway process can map it to a specific policy or chain of policies.
- **Static Content Provider:** You can use a Static Content Provider to serve static HTTP content from a particular directory. In this case, the Enterprise Gateway process is effectively acting as a web server.
- **Servlet applications:** The Enterprise Gateway process can act as a servlet container, which enables you to drop servlets into the HTTP Services configuration.

- **Packet Sniffer:** The Packet Sniffer is used to intercept network packets from the client, assemble these packets into complete HTTP messages, and log these messages to an audit trail. Because the Packet Sniffer operates at the network layer (unlike an HTTP-based traffic monitor at the application layer), the packets are intercepted transparently to the client. Because of this, the Packet Sniffer is a *passive service* that is typically used for SOA management and monitoring instead of general policy enforcement.

Organizing HTTP Services in HTTP Service Groups



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

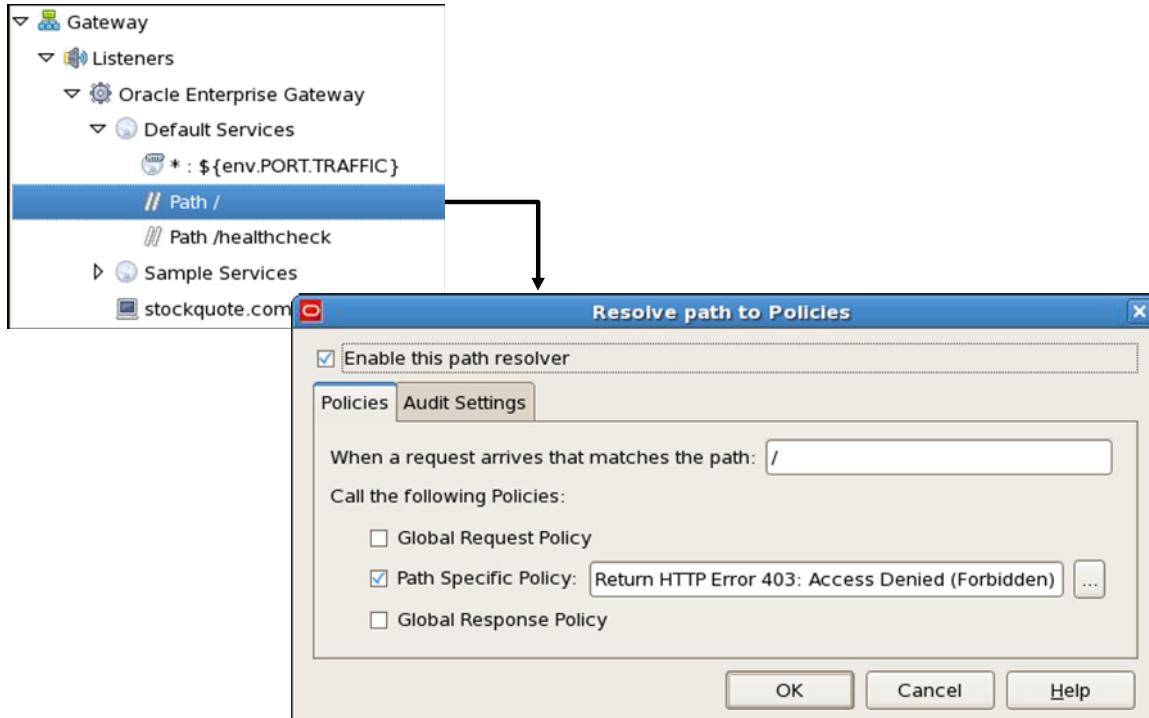
HTTP Service Groups

You can organize your HTTP services in the HTTP Services group. Usually, an HTTP Services group is configured for a particular type of HTTP service. For example, it would be typical to have an HTTP Interfaces group that contains the configured HTTP interfaces, and another Static Providers group to manage Static Content Providers. Though organizing HTTP services by type eases the task of managing services, the Enterprise Gateway is flexible enough to enable administrators to organize services into groups according to whatever scheme best suits their requirements.

By default, the Enterprise Gateway ships with three preconfigured HTTP Services groups (Default Services, Sample Services, and Management Services). By default, the Management Services node is not visible in Policy Studio, but it can be made visible by using the Options or Preferences menu in Policy Studio. The Default Services group contains some general-purpose default policies for use with an out-of-the-box installation of the Enterprise Gateway.

HTTP Services groups should consist of at least one HTTP interface together with at least one relative path. The HTTP interface determines which TCP port the process listens on, while you can use the relative path to map a request received on a particular path to specific policies. You can add several HTTP interfaces to the groups, in which case requests received on any one of the opened ports are processed in the same manner. For example, `http:// [HOST] :8080/AntiVirus` and `http:// [HOST] :8082/AntiVirus` requests can both be processed by the same policy (mapped to the `/AntiVirus` relative path).

Relative Path



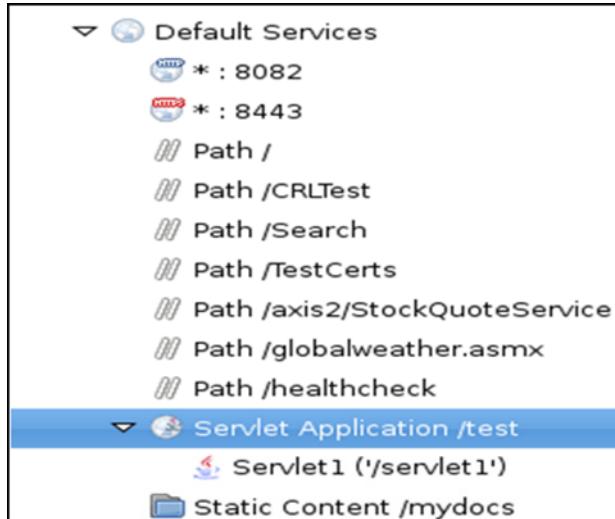
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

While HTTP and HTTPS Interfaces open sockets and listen for traffic, you can configure Relative Paths so that when a request is received on that path, the Enterprise Gateway process can map it to a specific policy or chain of policies.

Serving Static Content and Servlets

OEG comes with a web server and servlet server that can be used to host static content or servlets providing internal services.



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

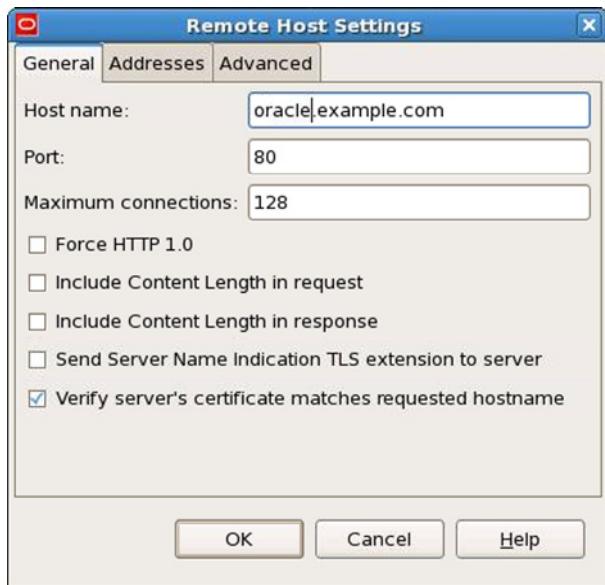
You can use the Static Content Provider in conjunction with an HTTP interface to serve static content from a specified directory. A relative path is associated with each Static Content Provider so that requests received on this path are dispatched directly to the provider and are not mapped to a policy in the usual manner.

For example, you can configure a Static Content Provider to serve content from the c:\docs folder (on a Windows system) when it receives requests on the relative path /docs.

Developers may want to write their own Java servlets and deploy them under the Enterprise Gateway to serve HTTP traffic. Conversely, they may want to remove some of the default servlets from the out-of-the-box configuration (for example, to remove the ability to view logging remotely). This pairing down of unwanted functionality may be required to further lock down the machine on which the Enterprise Gateway is running.

Note: Servlet applications should be used only by developers with very specific requirements and under strict advice from the Oracle Support team.

Adding a Remote Host



- Define a remote host when you need more control of the connections settings to a web services server.
- You must define a remote host if you want to track real-time metrics for this host.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A remote host definition can be added when you want to have more control of the connection settings to a particular web service server. The available connection settings include the following:

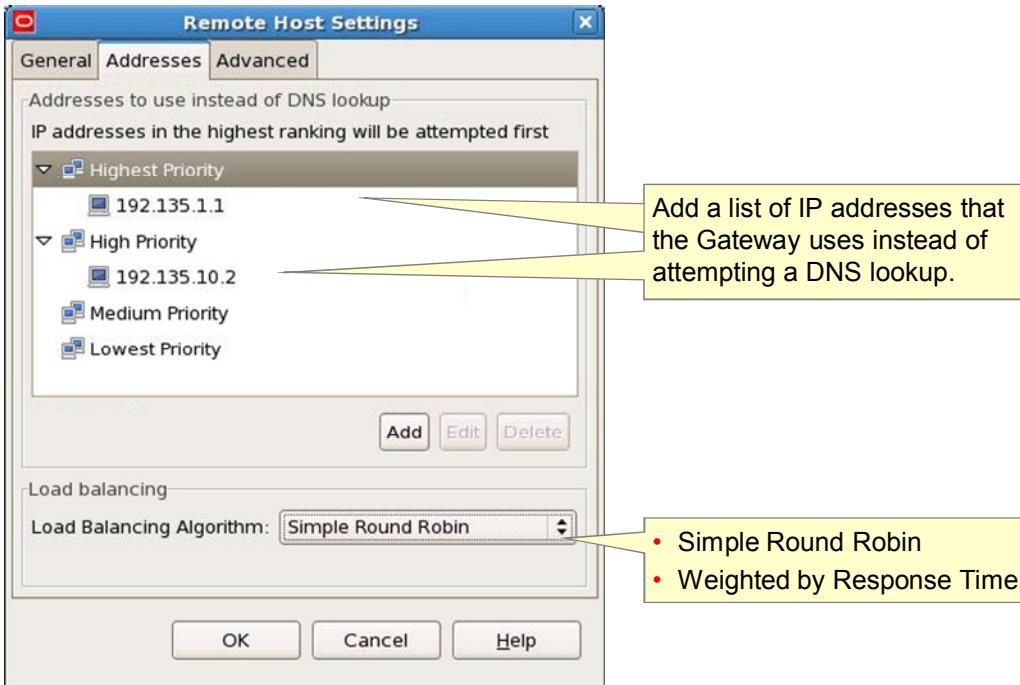
- HTTP version
- IP addresses
- Maximum number of connections to the remote host
- Timeouts (on the Advanced tab)
- Buffers (on the Advanced tab)
- Caches (on the Advanced tab)

Here are some use cases for configuring remote hosts with the Enterprise Gateway:

- Forcing the Enterprise Gateway to send only HTTP 1.0 requests to a destination server
- Mapping a host name to one or more specific IP addresses (for example, if a DNS server is unreliable or unavailable)
- Setting the timeout, session cache size, input/output buffer size, and other connection-specific settings for a destination server (for example, if the destination server is particularly slow, you can set a longer timeout)

You can add remote hosts for each process by right-clicking the process in the Policy Studio tree view and then selecting Add Remote Host.

Configuring Remote Host Settings



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can configure the following settings on the Addresses tab:

- **Addresses to use instead of DNS lookup:** You can add a list of IP addresses that the Enterprise Gateway uses instead of attempting a DNS lookup on the host name provided. This is useful in cases where a DNS server is not available or is unreliable. By default, connection attempts are made to the listed IP addresses on a round-robin basis.
- **Load balancing:** The Load Balancing Algorithm drop-down menu enables you to specify whether load balancing is performed on a simple round-robin basis or is weighted by response time. Simple Round Robin is the default algorithm. Connection attempts are made to the listed IP addresses on a round-robin basis in each priority group. The “Weighted by Response Time” algorithm compares the request/reply response times for the server address in each priority group. This is the simplest way of estimating the relative load of the address.

Quiz

An OEG policy can be mapped to:

- a. A process
- b. A relative path
- c. An HTTP service interface
- d. A remote host



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

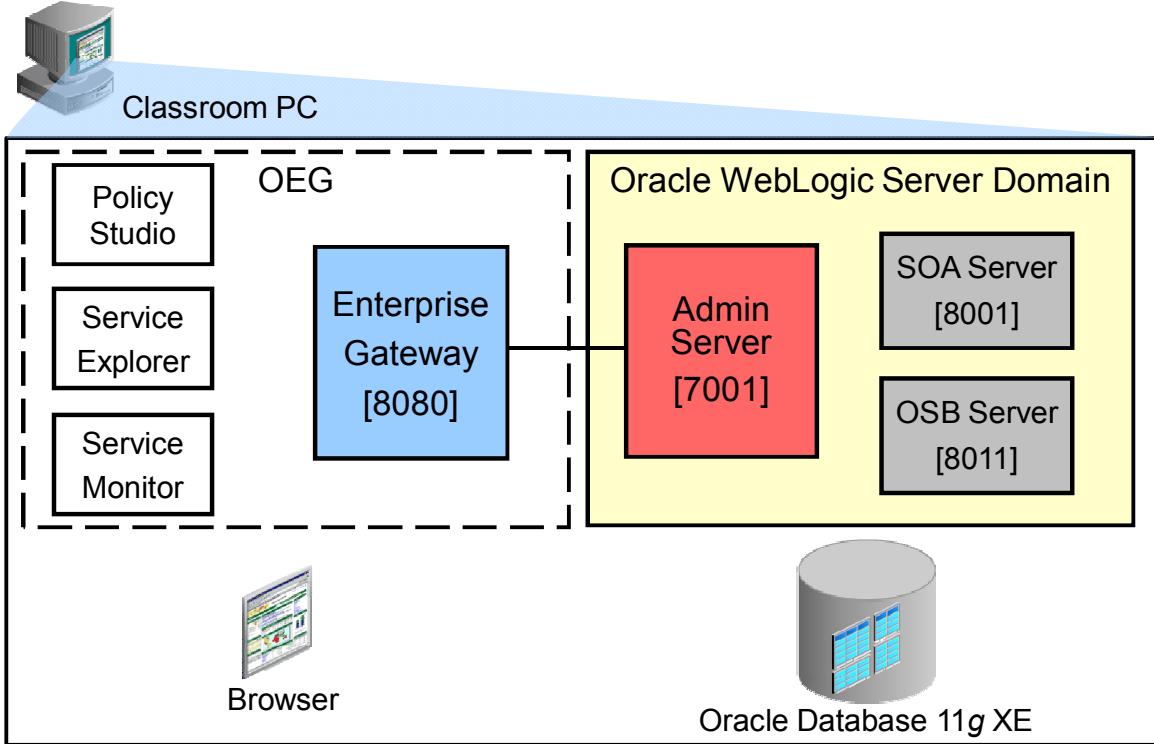
In this lesson, you should have learned how to:

- Describe the role Oracle Enterprise Gateway plays in a multilayered security deployment for web services
- Describe the capabilities of Oracle Enterprise Gateway
- Describe the Oracle Enterprise Gateway components
- Become familiar with the OEG interfaces
- Configure an Oracle Enterprise Gateway instance



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Lab Environment



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The student lab machine has OEG 11g, Oracle Database 11g XE, WebLogic Server 11g, Oracle SOA Suite 11g and Oracle Service Bus (OSB) 11g pre-installed and preconfigured as shown in the image. For most of the practices, you will use the OEG components and WebLogic Admin Server; SOA and OSB servers are needed only in the last practice (Practice 13).

Practice 3 Overview: Exploring the Gateway Configuration

This practice covers the following topics:

- Exploring the configuration of the Enterprise Gateway Server Instance
- Getting familiar with the user interface of Policy Studio



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Registering Web Services in OEG



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the capabilities of registered web services
- Outline the main steps of registering services in OEG
- Apply policies to the registered services



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Roadmap

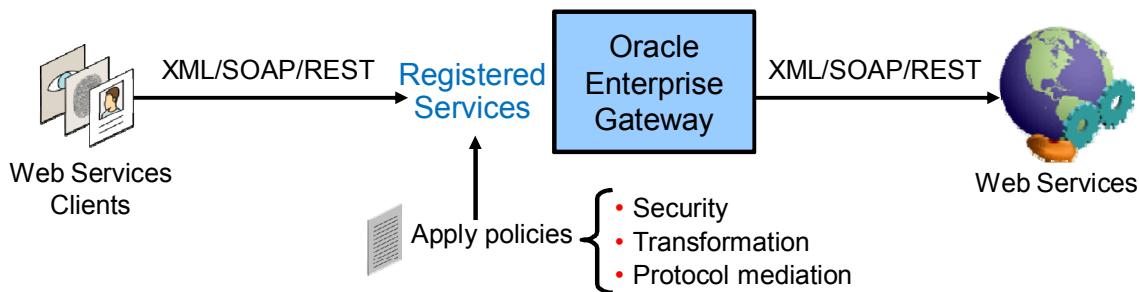
- Capabilities of registered web services
- Registering and testing a web service
- Applying policies to a registered web service



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Services Registration

- Exposes a virtualized version of the back-end web service to the clients
- Allows policies to be applied to the virtual services
- Enables many features of the gateway, including transformation, protocol mediation, and security
- Simplifies versioning, addressing, and SLA-based operations



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When you register a web service, the Enterprise Gateway exposes virtualized WSDL views of back-end target web services. The host and port for the web service are changed dynamically to point to the machine running the Enterprise Gateway. The client can then retrieve the WSDL for the virtualized web service from the Enterprise Gateway, without knowing its real location. For a registered web service, you specify the policy that the Enterprise Gateway enforces on the messages that it receives from the client. The Enterprise Gateway allows policies to be applied to virtual services, not to the back-end services themselves.

As part of web service lifecycle management, the Enterprise Gateway uses the virtualized version of web services to simplify versioning, addressing, and operations based on service-level agreements (SLAs).

Capabilities of Registered Web Services

OEG provides the following capabilities to the registered web services:

- XML firewalling
- Enriching messages
- Routing and offloading
- Throttling
- Converting protocols

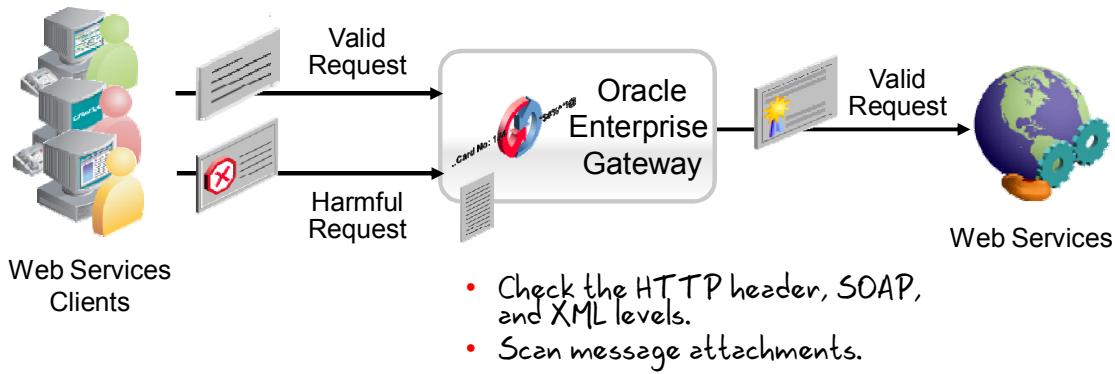


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This section discusses the capabilities provided by OEG to the registered web services.

XML Firewalling

- Problem:
Web services are being exposed to the public Internet.
Incoming data may contain threatening content.
- Solution:



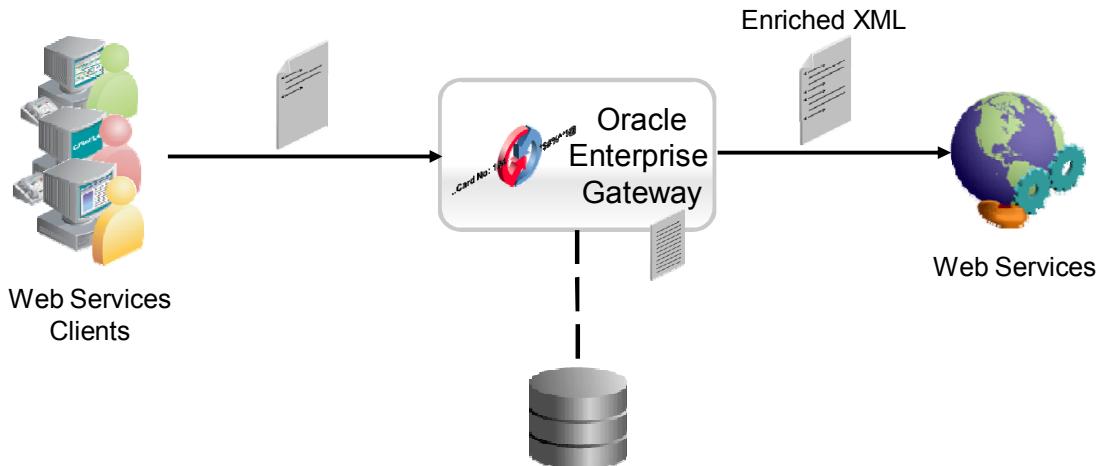
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Enterprise Gateway typically resolves an incoming message to a specific target web service by examining either the SOAP message header or, with native XML, the HTTP header, and the message body with its fields, parameters, and attachments. After the target web service is resolved, the XML firewall can apply a stored security policy based on the target address, originating caller identity, and message content to block the harmful messages.

Enriching Messages

- Problem:
A web services application requires information that cannot be passed in the incoming message.
- Solution:



ORACLE®

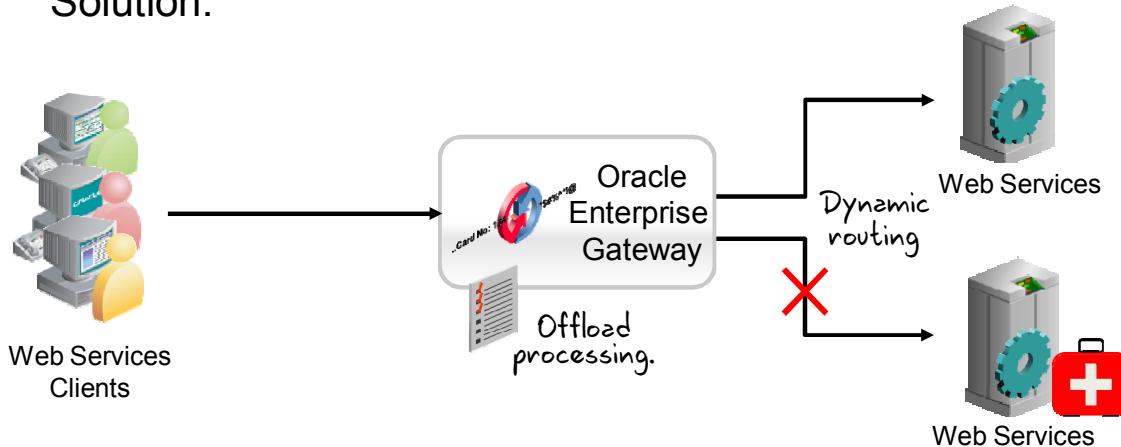
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

If a web service requires customer information that cannot be passed in the incoming message, the gateway can look up the customer information from the database and then “inject” data into XML and/or message headers. Meanwhile, the Enterprise Gateway can also cache the responses from the database and inject data into the next request with the same situation.

By putting this functionality on the XML network infrastructure, data is automatically populated in XML messages before they reach the consuming web services. This actually simplifies and accelerates applications in application servers.

Dynamic Routing and Offloading

- Problem:
If a service is running slowly, it is important for the gateway to take remedial action.
- Solution:



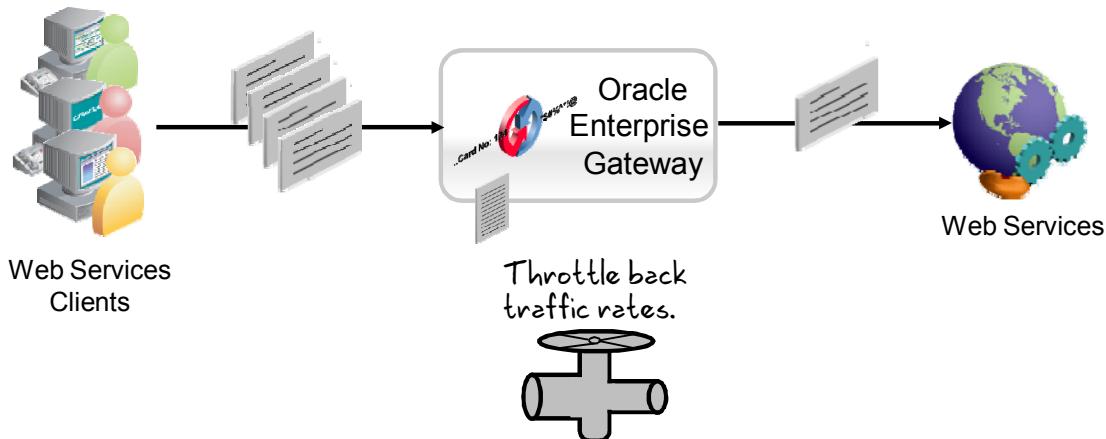
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The gateway performs “active management” by dynamically routing traffic to other endpoints when one endpoint is running slowly. If services are not meeting their service-level agreement (SLA), the gateway can dynamically offload processing such as XML Schema Validation and XSLT from the service.

Throttling

- Problem:
A web service is brought down due to excessive traffic.
- Solution:



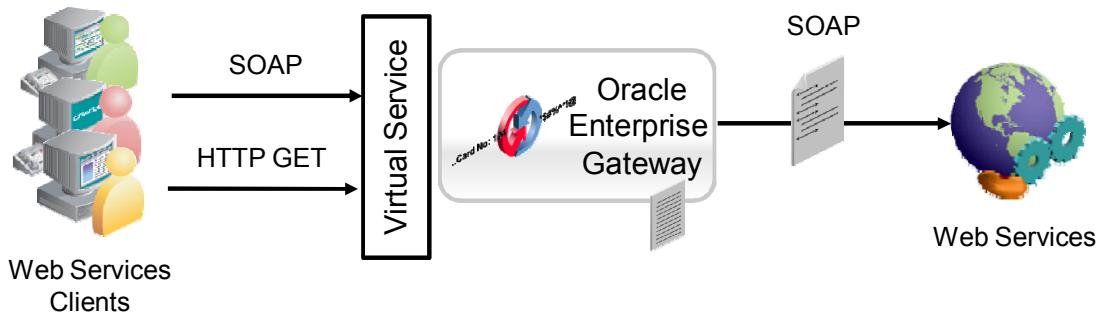
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

If a web service is experiencing excessive traffic, you can use the Enterprise Gateway to throttle back traffic rates to what the web service can handle, thereby smoothing the traffic “spikes.” So the web service conforms to the SLA configured in the Enterprise Gateway.

Converting Protocol: REST and SOAP

- Problem:
An organization has deployed SOAP web services, but its customers want to use REST interfaces instead.
- Solution:



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Suppose that you have SOAP web services, but your customers ask for REST web services instead. Because it is easier to write a REST client, the messages are smaller, and you can cache REST traffic by using standard web infrastructure.

OEG enables you to create REST web services in front of SOAP web services (as shown in the slide). The REST service in this example is a “virtual service” because it does not actually exist on the web service platform itself. Only the SOAP service exists there. Both SOAP and REST interfaces are exposed in front of the same back-end SOAP service.

Quiz

OEG policies are applied to the registered web services in the Enterprise Gateway.

- a. True
- b. False



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a

Roadmap

- Capabilities of registered web services
- Registering and testing a web service
- Applying policies to a registered web service



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Registering a Service in Enterprise Gateway

You can register a service by using two tools:

- Service Manager
- Policy Studio

Service Manager	Policy Studio
Web-based	Eclipse RCP-based
Supports creating new policies by chaining existing policies	Comprehensive policy editor
Enables you to register a service and apply policies at specific interception points	

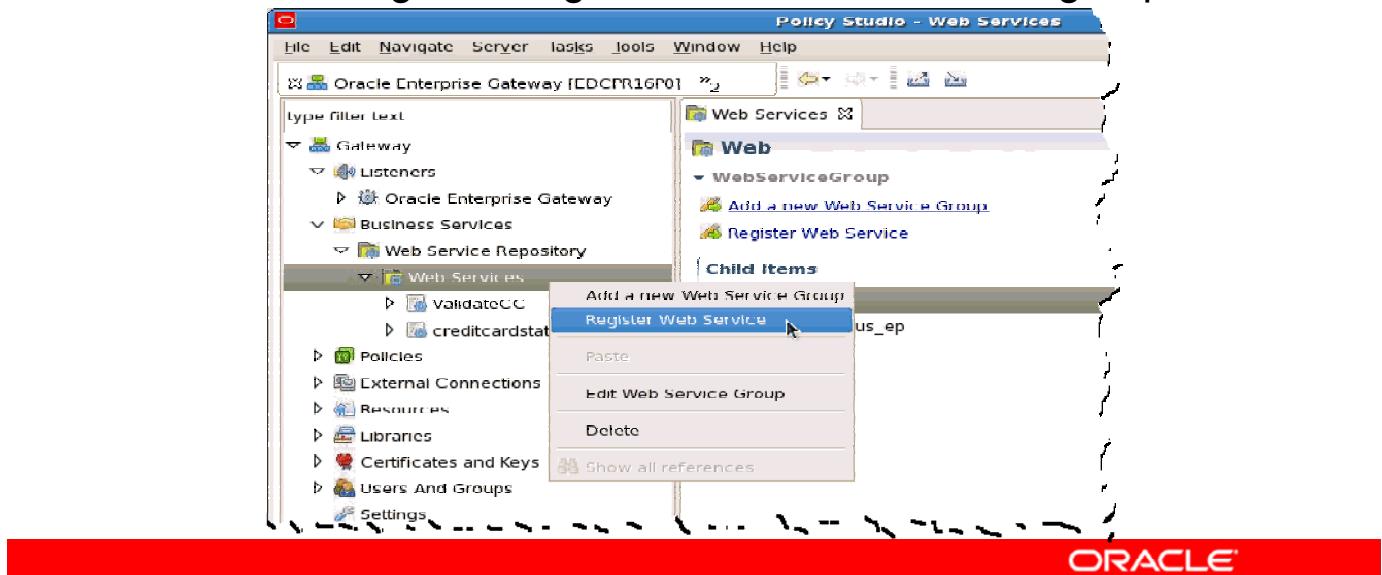


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can register a web service by using either Service Manager, a web-based system administration tool, or Policy Studio, an Eclipse RCP-based authoring tool. Comparing to Policy Studio, a comprehensive policy editor, Service Manager has less functionality, but provides quick and easy access to enable you to manage the web services and policies online.

Registering Services in Policy Studio

- You register web services by importing the WSDL file.
- Web services are registered into the Web Services Repository.
- You can organize registered web services into groups.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To register web services in Policy Studio, you import the WSDL file that contains the web services definitions into the Web Services Repository, which is displayed as a top-level node in the Policy Studio Policies tree view.

WSDL files can be imported from the file system, URL, or UDDI registry, and organized in web service groups, which provide a convenient way of keeping groups of related web service definitions together. You can register the same service multiple times as long as it is placed in a different group and you override the relative path that is used to call the service.

You can also test the WSDL file for compliance with Web Services Interoperability (WS-I) standards.

Steps to Register Services

1. Specify where to retrieve the service definition (WSDL file, URL, UDDI).
2. Specify the operations to be exposed by the gateway.
3. Specify whether the nonselected operations should be removed from the WSDL exposed to clients.
4. Specify the port from which the service will be accessible.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Policy Studio provides a wizard to help you register a web service. The main steps you need to follow in the wizard are listed in the slide.

Note that, in step 2, the same service back-end can be exposed multiple times, with different operations mapping to different virtual services.

What Is Created?

Registration has the following results:

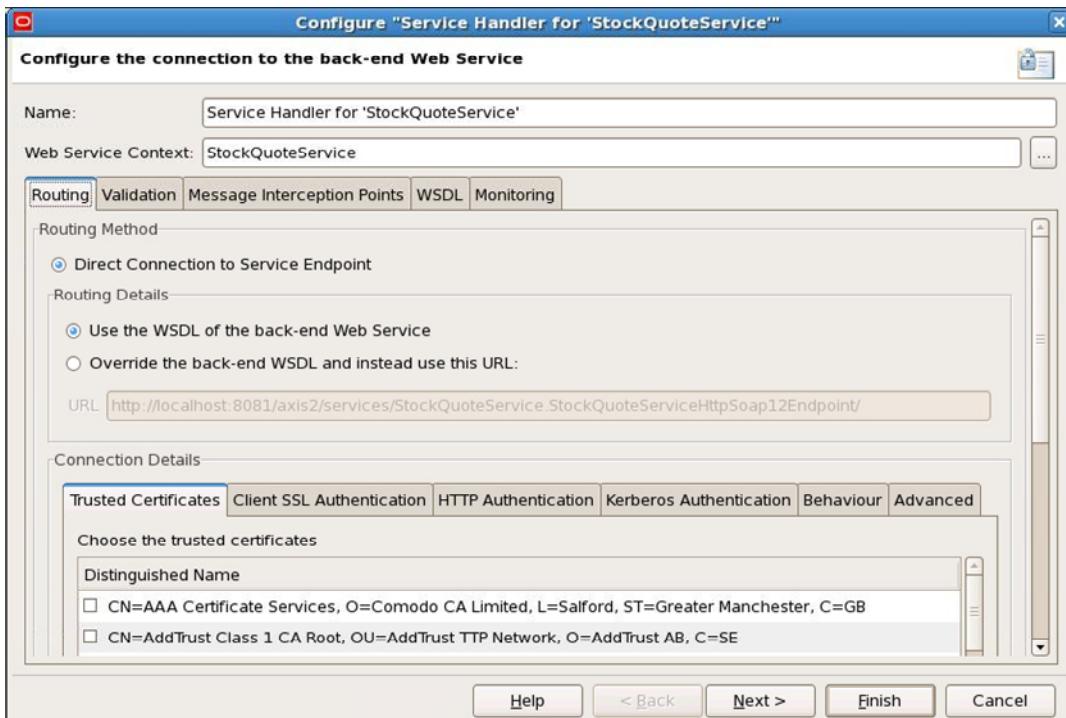
- A Service Handler object is generated.
- The ServiceName is extracted from the service definition file and used for monitoring.
- A web service context is created on the port to expose the service to the gateway clients.
- The schemas from the WSDL file (or referenced from the WSDL file) are imported in the configuration and used for XML validation.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Importing WSDL files into the repository will auto-generate a Service Handler that is used to control and validate requests to the web service and responses from the web service. The information that is used to configure the Service Handler is automatically taken from the operation definitions in the WSDL.

Service Handler



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A Service Handler is automatically generated when the service is registered. It is the single-stop filter used to control and validate requests to the web service and responses from the web service. You can configure the following settings:

- **Routing:** When routing to a service, you can specify a direct connection to the web service endpoint by using the URL in the WSDL, or you can override this URL by entering a URL in the field provided. Alternatively, in cases where the routing behavior is more complex, you can delegate to a custom routing policy that takes care of the added complexity. These alternative routing configurations are available on the Routing tab.
- **Validation:** When using a WSDL to import a service into the Web Services Repository, Policy Studio can extract the XML Schema from the WSDL and configure the Enterprise Gateway to validate incoming requests against it. Select Use WSDL Schema if you want to validate incoming requests against the Schema in the WSDL. Alternatively, you can create a custom-built policy to validate the contents of incoming requests.

- **Message Interception Points:** The settings on the Message Interception Points tab enable you to configure how the request and response messages for the service are processed as they pass through the Enterprise Gateway. Several message interception points are exposed to enable you to connect to different stages of the Enterprise Gateway's request processing cycle.
At each of these interception points, it is possible to run policies that are specific to that stage of the request processing cycle.
- **WSDL:** The WSDL tab enables you to define whether this service should be exposed via WSDL to clients. It is important to note that the exposed WSDL represents a virtualized view of the back-end web service.
- **Monitoring:** The fields on this tab enable you to configure whether this web service stores usage metrics data to a database. This information can then be used by Service Monitor to produce reports showing how this web service is being called and who is calling it.

Registering Services in Service Manager

Service Manager can be accessed at the following URL:

<http://localhost:8090/manager/>

The screenshot shows the Oracle Service Manager interface. On the left, the 'Web Services' section displays a list of services: 'StockQuoteService' (selected), 'ValidateCC', and 'creditcardstatus_ep'. Each service has a 'WSDL' link next to it. Below this is an 'Edit Web Service' panel for 'StockQuoteService' with fields for 'Name' (StockQuoteService), 'Port' (8082), and 'WSDL Url' (http://host01.example.com:8082/axis2/services/StockQuoteService_StockQuoteServiceHttpSoap12Endpoint?WSDL). On the right, the 'Policies' section lists various policies under 'Policy Library' and 'WS-Policy' categories, such as AV, HTTPBasicToSAML, Health Check, etc.

ORACLE

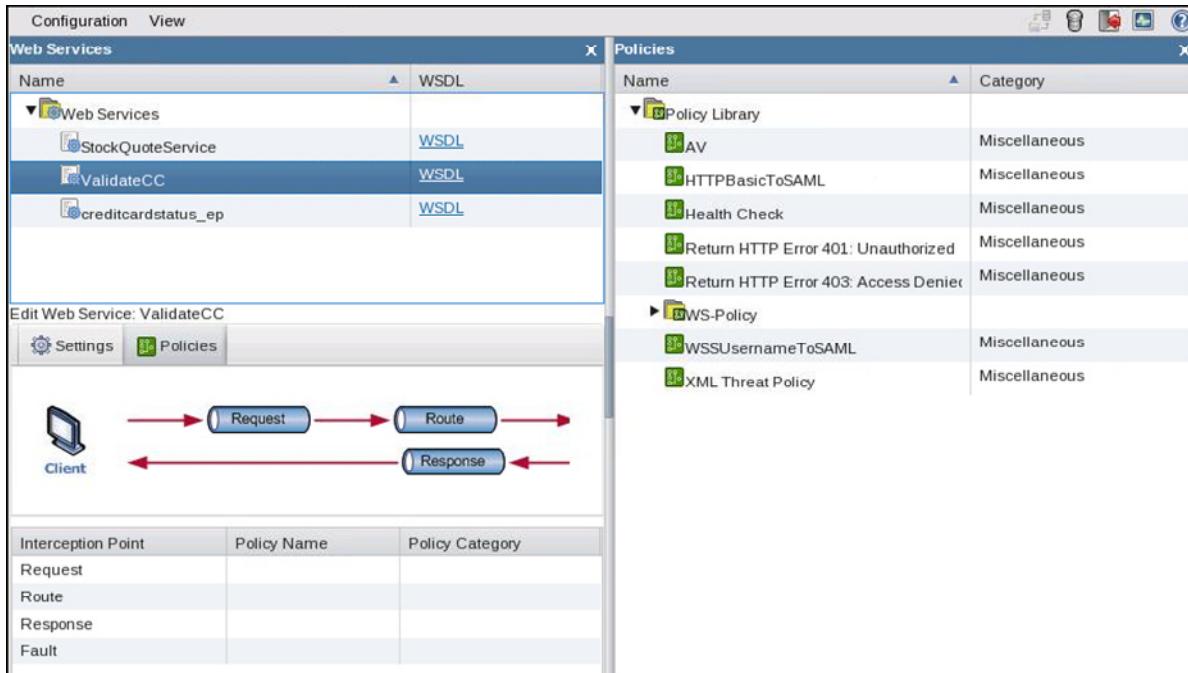
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Service Manager is a web-based system administration tool that simplifies Enterprise Gateway management tasks. It provides quick and easy access to enable you to manage your web services and policies online. Only gateway administrators have access to Service Manager, with which they can perform the following tasks:

- Register web services and assign policies.
- Add policies composed of existing policies.
- Deploy configuration to the Enterprise Gateway.
- Access real-time monitoring of web services and messages.

After a service is registered, you can see the service settings in the UI and reach the client WSDL (exposed by the gateway).

Applying Policies in Service Manager

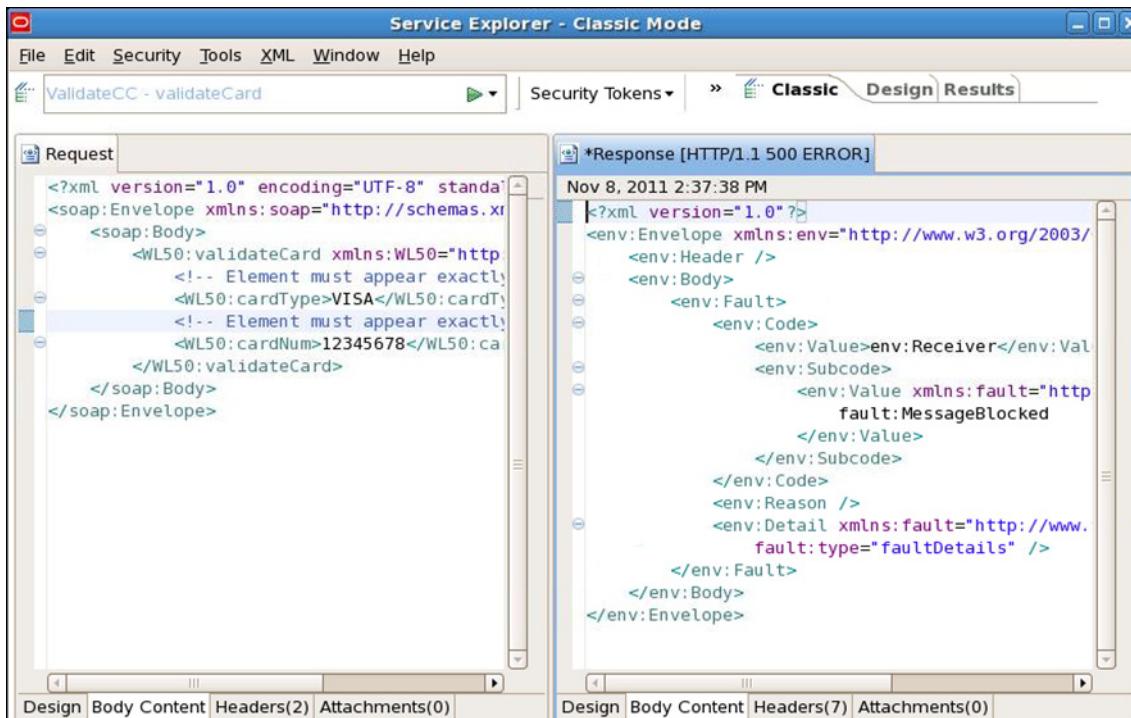


ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can use Service Manager to apply policies to the registered service by simply dragging policies from the library to the right onto the different interceptions points (Request, Response, Route, and Fault).

Testing Registered Services by Using Service Explorer



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can use Service Explorer to test the protected web service by sending a message through the Enterprise Gateway.

Quiz

Both Policy Studio and Service Manager allow you to:

- a. Register web services
- b. Assign policies to virtualized services
- c. Edit the policies
- d. Create policies composed of existing policies



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, b, d

Roadmap

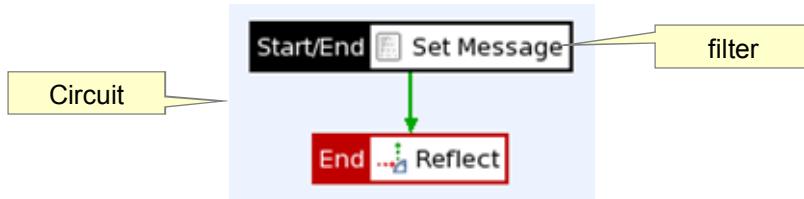
- Capabilities of registered web services
- Registering and testing a web service
- Applying policies to a registered web service



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Policy Concepts

- Policy: A network of message filters in which each filter is a modular unit that processes a message
- Filter: An executable rule that performs a specific type of processing on a message
- Circuit: A series of filters through which a message passes



ORACLE®

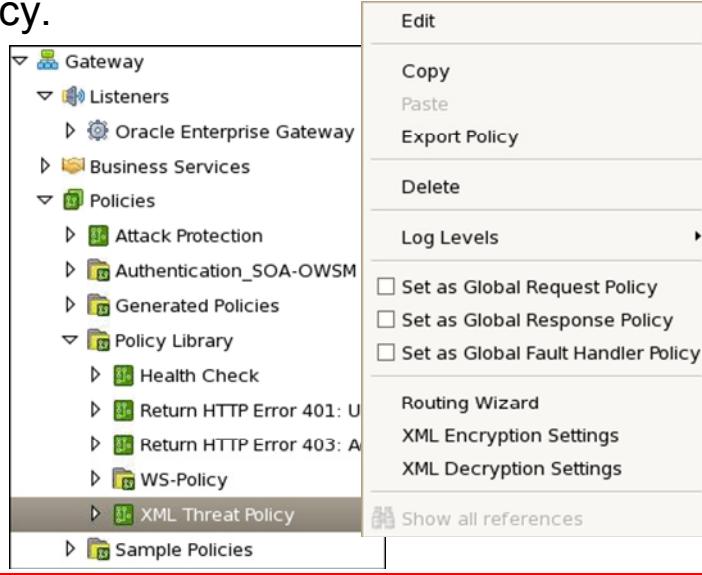
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- **Policy:** A policy is a network of message filters in which each filter is a modular unit that processes a message. A message can traverse different paths through the network, depending on which filters succeed or fail. For example, this enables you to configure policies that route messages that pass a Schema Validation filter to a back-end system, and route messages that pass a different Schema Validation filter to a different system.
- **Filter:** A filter is an executable rule that performs a specific type of processing on a message. For example, the Message Size filter rejects messages that are greater than or less than a specified size. There are many categories of message filters available with the Enterprise Gateway, including authentication, authorization, content filtering, signing, and conversion. In Policy Studio, a filter is displayed as a block of business logic that forms part of an execution flow known as a *circuit*.

- **Circuit:** A circuit is a series of filters through which a message passes. A circuit can contain other circuits, which enables you to build modular reusable policies. In Policy Studio, a circuit is displayed as a path through a set of filters. A filter can have only one success path and one failure path. You can use these success paths and failure paths to create sophisticated rules.

Global Policies

- Global Policies can be applied at the request/response level.
- A specific policy can be assigned to a service to override the global policy.



ORACLE

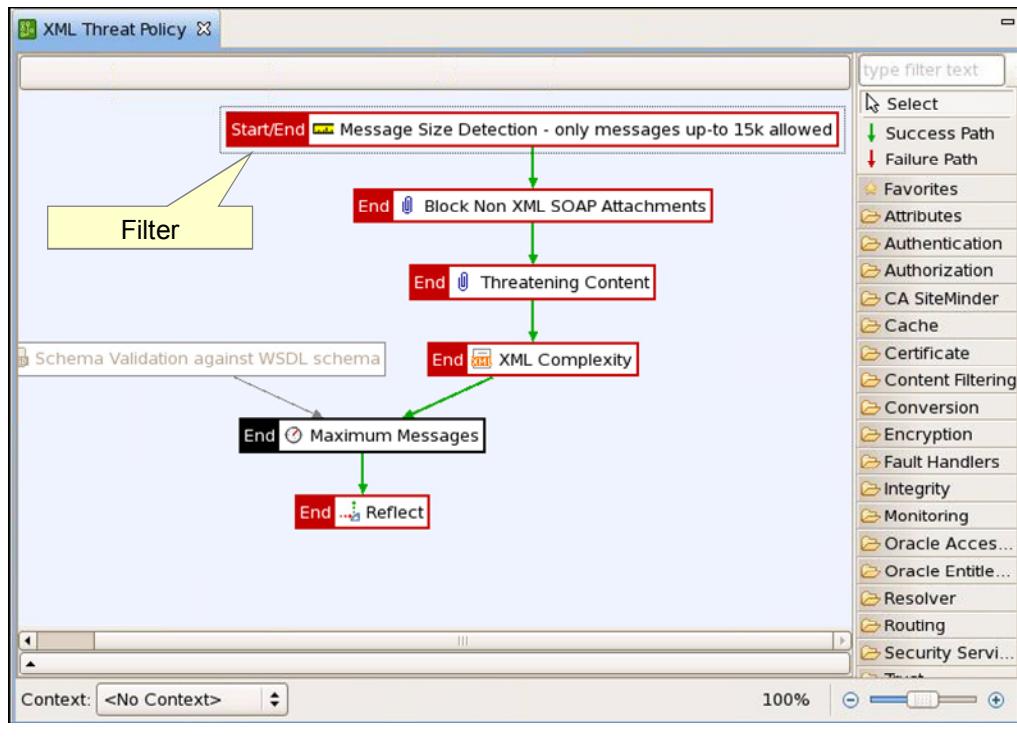
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Global policies enable you to label policies with specific roles in the Enterprise Gateway configuration. For example, you can label a specific policy such as XML Threat Policy as a Global Request policy. This policy can be executed globally on the request path for all messages passing through the Enterprise Gateway. Using a global policy in this way enables you to use the same policy on all requests, and for multiple services. It also means that you can change the labeled global policy to a different policy without needing to rewire any existing policy circuits.

To set a global policy, right-click any policy and choose the option as per the screenshot in the slide.

To find the global policies in the policy library, right-click the top level Policies container and choose Show Global Policies...

Getting Started with Policies



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

OEG policies are implemented by designing circuits that chain the execution of filters.

Green and red arrows define transitions to the next step in case of successful and nonsuccessful execution of a filter. You can have only one transition of each type for each filter. The screenshot shows an example circuit with success paths.

A circuit must have a starting filter (in this example, it is Message Size Detection). Filters that are marked with the word End will end the execution of the policy if the filter execution fails.

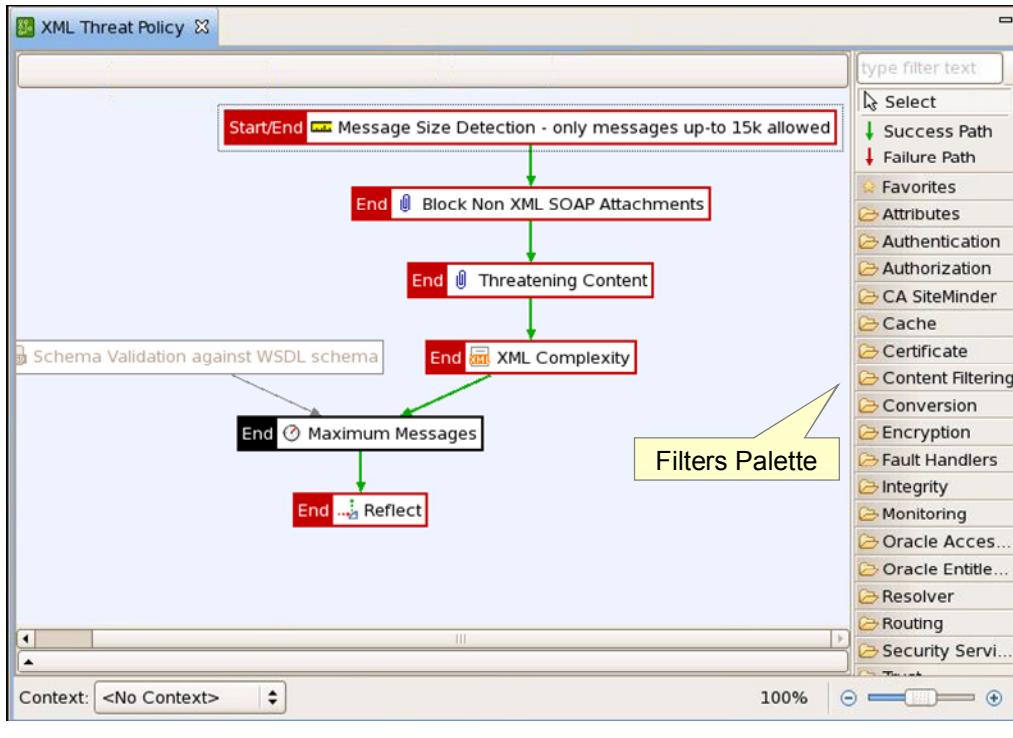
Filters marked with Start/End indicate that:

- The circuit execution starts here
- The circuit stops executing if this filter fails

Note: A circuit with a single filter marked with Start/End is perfectly valid.

Policies can execute on their own or as part of another policy (allowing reuse). As an example, the “Checking against threats” filter calls another policy.

Filters Palette



ORACLE

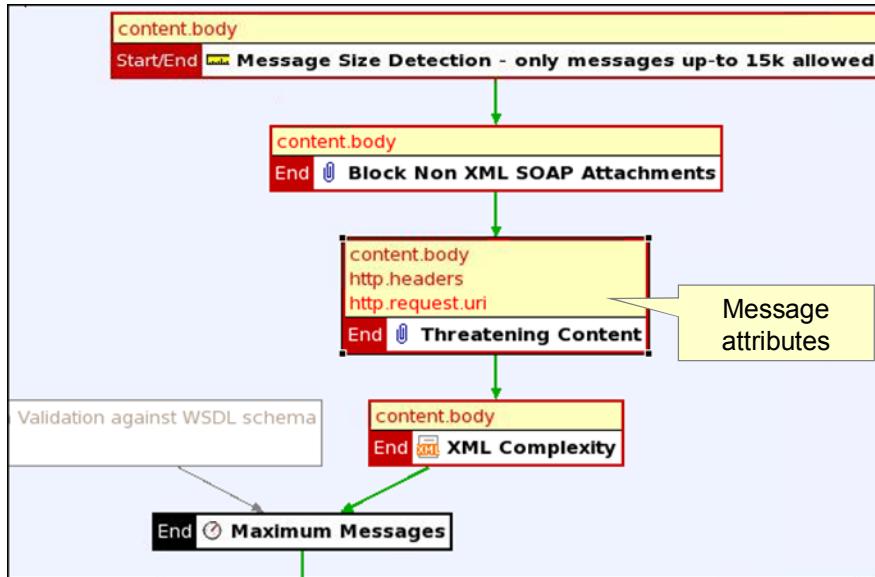
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Policy Studio provides a searchable list of numerous filters on the Filters Palette. These filters are classified by category. You can click and drag a filter to a policy to use it.

Drop the filter:

- On an existing transition (arrow) to place the filter between two existing ones
- On another filter to automatically create a transition between the existing filter and the new one
- On the canvas to connect freely to another part of the circuit

Filters and Message Attributes



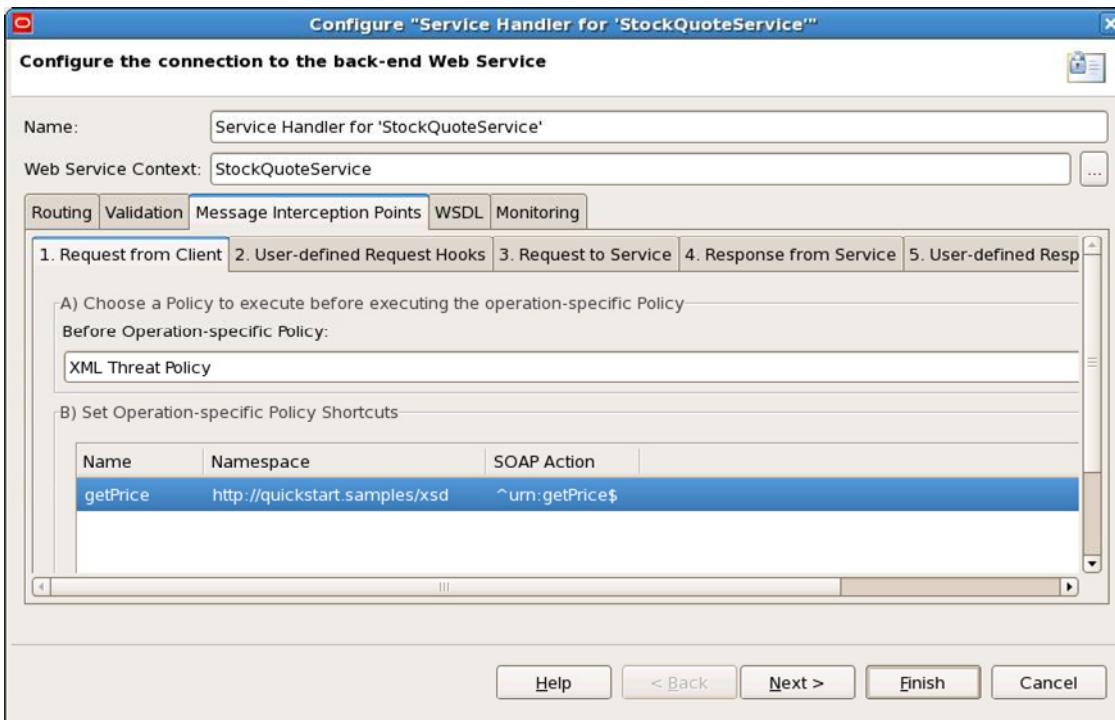
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Each filter requires input data and produces output data. This data is stored in message attributes, and you can access their values by using syntax such as \${attribute.name}. You can also use specific filters to create your own message attributes and to set their values. The full list of message attributes flowing through a policy is displayed when you right-click the Policy Studio canvas and select Show All Attributes. You can also hover your mouse over a filter to see its inputs and outputs. The Trace filter enables you to trace message attribute values at execution time.

If a filter needs an attribute as input, which has not been generated in the previous executions steps, it will be flagged in red. This may be an indication of a problem that you need to investigate (chaining of filters or policies) or an indication that the policy cannot run on its own.

Applying Policies to Registered Web Services



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

On the Message Interception Points tab, you can apply policies that are specific to that stage of the request processing cycle. For example, you can configure a logging policy to run just before the request has been sent to the web service and then again just after the response has been received.

Typically, the configuration settings on this tab are automatically configured when importing a service into the Web Services Repository based on information contained in the WSDL. In cases where the WSDL contains WS-Policy assertions, a number of policies are automatically generated and connected to perform the relevant security operations on the message.

You can apply policies at six interception points. The policies are executed in the following order:

1. **Request from Client:** The first message interception point enables you to run a policy against the request as it is received by the Enterprise Gateway. Typically, this is where authentication and authorization events should occur.
2. **User-defined Request Hooks:** Users should use this interception point primarily to connect their own custom-built request processing policies.

3. **Request to Service:** This enables you to alter the message before it is routed to the web service. For example, if the service requires the message to be signed and encrypted, you can configure the necessary policies here.
4. **Response from Service:** This is executed on the response returned from the web service.
5. **User-defined Response Hooks:** You should use this interception point primarily to connect custom-built response processing policies.
6. **Response to Client:** This enables you to process the response before it is returned to the client.

Quiz

Which of the following statements is NOT true concerning policies:

- a. Policies can execute on their own or as part of another policy.
- b. A policy is a chain of execution of filters.
- c. A circuit with a single filter marked with Start/End is not valid.
- d. A filter can have only one success path and one failure path.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Summary

In this lesson, you should have learned how to:

- Describe the capabilities of registered web services
- Outline the main steps in registering services using OEG
- Apply policies to the registered services



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 4 Overview: Registering Web Services in the Gateway

This practice covers the following topics:

- Registering the web service
- Testing the registered web service in Service Explorer



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Monitoring, Logging, and Tracing

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Monitor traffic by using OEG monitoring tools
- Identify the differences between OEG logging and tracing
- Configure log settings
- Set tracing levels
- View trace information



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Troubleshooting Tools

Tools to help troubleshoot common issues:

- Real-time monitoring
- Verbose logging capability (Logging/Audit Trail)
- Using Log Payload Filter
- Trace output



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

OEG ships with features that can help determine and remediate issues more easily when they occur. The slide lists the tools that the product provides for identifying issues.

- The traffic monitor gives you a live and graphical view of transactions going through the gateway.
- Logging/Audit trails are very handy for tracking and analyzing what happens to a message that is being processed through the gateway by keeping a record of every transaction. With logging, it is possible to add “log points” to the audit trail and the content of these logs points is also customizable.
- The trace files created by OEG provide valuable information about where a message request or other issues could have occurred.

Roadmap

- Monitoring traffic
- Setting tracing levels
- Configuring log settings

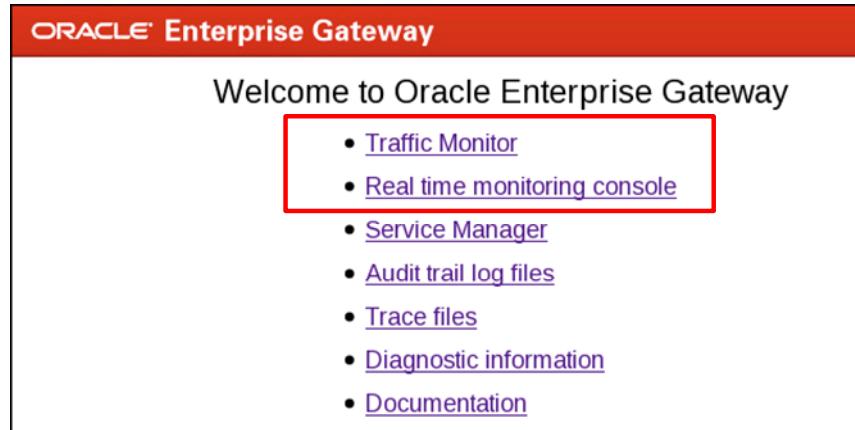


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Monitoring

OEG provides three monitoring tools:

- Traffic Monitor: Viewing the message log of the HTTP and HTTPS traffic processed by the Enterprise Gateway
- Real-time monitoring console: Monitoring messages
- Service Monitor: Viewing historical traffic data and analyzing service usage



ORACLE

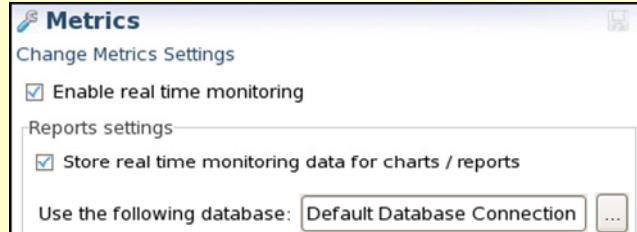
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Enterprise Gateway provides the following monitoring tools:

- **Real time monitoring console:** A real-time monitoring tool that enables you to monitor messages processed by the Enterprise Gateway
- **Traffic Monitor:** Provides a web-based message log of the HTTP and HTTPS traffic processed by the Enterprise Gateway. The Traffic Monitor tool is used for operational diagnostics.
- **Service Monitor:** A separately installed component that enables you to generate reports on the traffic history of multiple Enterprise Gateways and analyze the service usage

Note: Enabling monitoring has a negative impact on performance. If you want to maximize performance, do not enable these settings.

Enabling Monitoring

Tool	Enabling Setting
Traffic Monitor	 A screenshot of the 'Traffic' settings dialog box. It shows a title bar with 'Traffic' and a sub-section titled 'Change Traffic Monitor'. Below this is a checkbox labeled 'Enable Traffic Monitor' which is checked.
Real-time monitoring	 A screenshot of the 'Metrics' settings dialog box. It shows a title bar with 'Metrics' and a sub-section titled 'Change Metrics Settings'. Below this is a checkbox labeled 'Enable real time monitoring' which is checked.
Service Monitor	 A screenshot of the 'Metrics' settings dialog box, similar to the one above but with additional sections. It includes a 'Reports settings' section with a checkbox 'Store real time monitoring data for charts / reports' which is checked. At the bottom, there is a field 'Use the following database:' followed by a dropdown menu set to 'Default Database Connection'.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You must enable the monitoring tools in Policy Studio before viewing traffic statistics or collecting data.

- To enable traffic monitoring in Policy Studio:
 1. On the Services tab on the left, right-click the Oracle Enterprise Gateway process, and select Monitoring > Traffic.
 2. In the Traffic Monitor Settings dialog box, select Enable Traffic Monitor.
- To enable real-time monitoring, you must first enable traffic monitoring (as described above) and real-time monitoring:
 1. On the Services tab on the left, right-click the Oracle Enterprise Gateway process, and select Monitoring > Metrics.
 2. In the "Real time monitoring settings" dialog box, make sure that "Enable real time monitoring" is selected.

- For Service Monitor to be able to generate usage reports, you need to enable real-time monitoring and configure the Enterprise Gateway to store message metrics in the same database from which Service Monitor is configured to read. Perform the following steps:
 1. On the Services tab on the left, right-click the Oracle Enterprise Gateway process, and then select Monitoring > Metrics.
 2. In the “Real time monitoring settings” dialog box, make sure that “Store real time monitoring data for charts/reports” is selected.

Setting Up Database for Service Monitor Reports

- Creating the database and adding the database user
- Setting up database tables by using the SQL script `db_schema.sql`, located in the `INSTALL_DIR/system/conf/sql/DB_Name` folder

Run the SQL commands in the `db_schema.sql` file:

```
SQL> connect oeguser/welcome1
Connected.
SQL> @/u01/oracle/oeg11g/oegservicemonitor/
      system/conf/sql/oracle/db_schema.sql
```

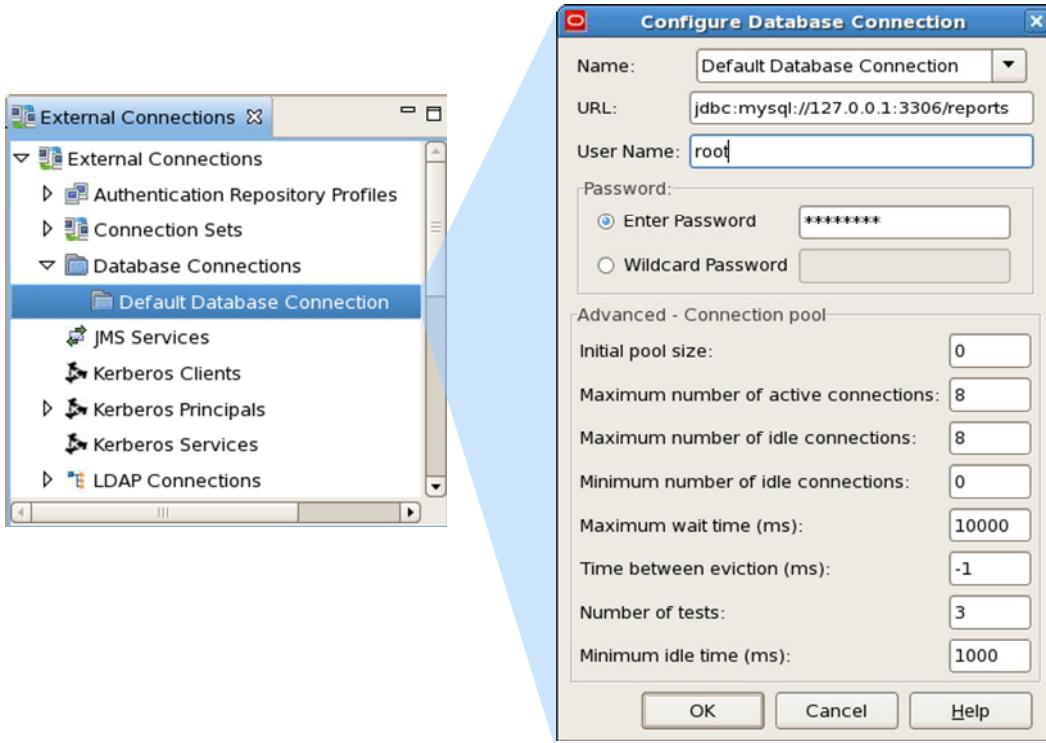


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Enterprise Gateway stores and maintains the audit trail and message metrics data in a JDBC-compliant database, from which Service Monitor reads data and generate reports. So you first need to create the database of your choice.

OEG-supported database products include MySQL, Microsoft SQL Server, Oracle, and IBM DB2. After the database is created, you can then set up the relevant database tables. Service Monitor provides SQL scripts to set up the database tables for each of the supported databases. The scripts are provided in the `INSTALL_DIR/system/conf/sql` folder, in each database's subdirectory.

Configuring Database Connection for Enterprise Gateway



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When the reports database table is created, you need to configure the Enterprise Gateway in Policy Studio to store message metrics in the database. The database connection is configured in the External Connections interface. A default database connection is populated with the connection details to MySQL database. You can either edit the details for the default database connection or add a new connection to connect to the database in which you intend to store the metrics.

The following table lists examples of connection URLs for the supported databases, where `reports` is the name of the database and `DB_HOST` is the IP address or host name of the machine on which the database is running:

Database	Example Connection URL
Oracle	<code>jdbc:oracle:thin:@DB_HOST:1521:reports</code>
MS SQL	<code>jdbc:sqlserver://DB_HOST:1433;DatabaseName=reports;integratedSecurity=false;</code>
MySQL	<code>jdbc:mysql://DB_HOST:3306/reports</code>
IBM DB2	<code>jdbc:db2://DB_HOST:50000/reports</code>

Monitoring Traffic

The screenshot shows the Oracle Enterprise Gateway Traffic Monitor interface. At the top, the title bar reads "Traffic Monitor - Mozilla Firefox". Below the title bar is a menu bar with File, Edit, View, History, Bookmarks, Tools, and Help. The main content area has a header "Message Traffic Log" and a sub-header "Search results for: last 100". A table lists log entries:

*	Method	Status	Path	Service	Operation	Subject	Date/Time
●	POST	200	/validatecc/ValidateCCPort	ValidateCC	validateCard		10/2/11 10:29 AM
●	POST	500	/validatecc/ValidateCCPort	ValidateCC	validateCard		10/2/11 10:29 AM
●	POST	200	/val				
●	POST	200	/val				
●	POST	200	/val				

A modal window titled "Transaction Details" is open, showing a transaction ID: 5e170dbd4eb11b9006d90000. It includes a "Filter execution path" section with a tree view:

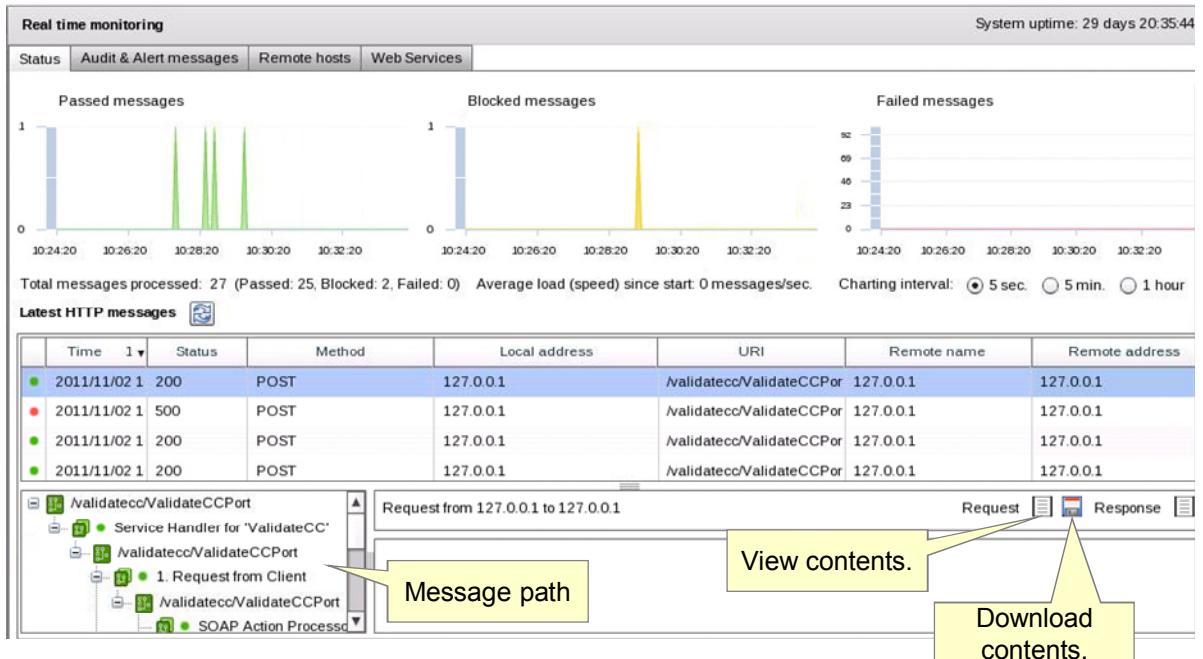
- Filter
- ✓ Validatecc/ValidateCCPort
- ✓ Service Handler for 'ValidateCC'
 - ✓ Validatecc/ValidateCCPort
 - ✓ 1. Request from Client
 - ✓ /validatecc/ValidateCCP

At the bottom of the interface, there is a red footer bar with the Oracle logo and the text "Copyright © 2012, Oracle and/or its affiliates. All rights reserved."

Click the Traffic Monitor link on the “Welcome to Oracle Enterprise Gateway” page to view the Message Traffic Log, which provides a historical log of the HTTP and HTTPS traffic processed by the Enterprise Gateway. You can filter messages on a range of criteria. The default filters include transaction ID, message from (client or gateway), maximum results, and time. The Settings button (on the top right of the screen) enables you to change logging settings as and when needed. You do not need to refresh or deploy to the Enterprise Gateway. You can specify the log settings at the system level and the HTTP interface level.

Click a message entry in the Message Traffic Log to view transactions details for that message. The transaction details include the filter execution path, the contents of the outbound and inbound HTTP request and response messages, and the message trace.

Real-Time Monitoring



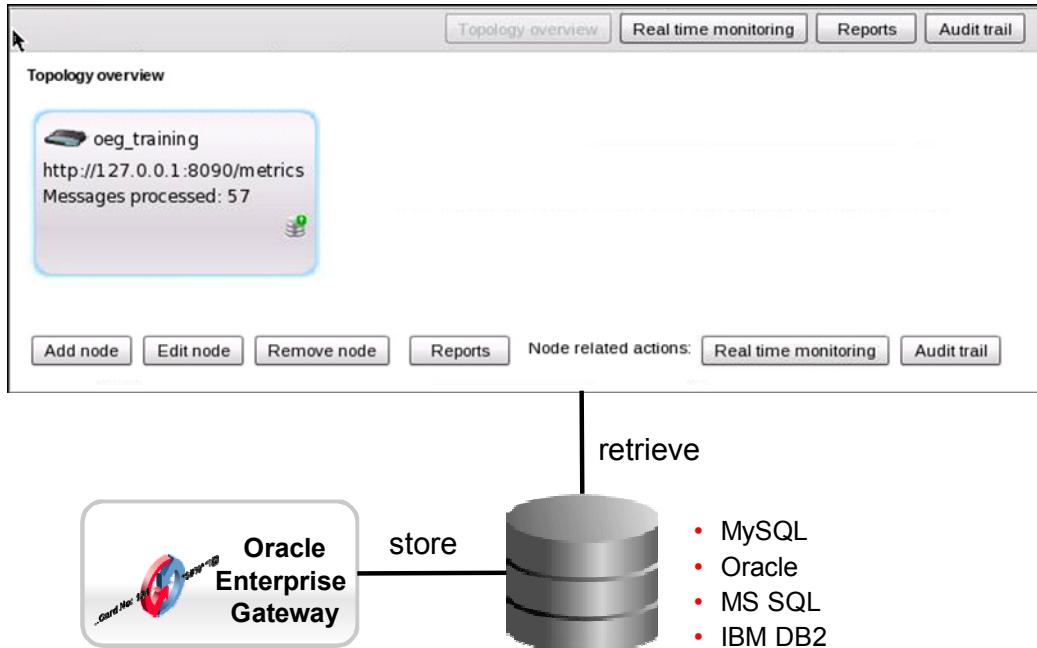
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can view a range of monitoring statistics in the Real time monitoring console (for example, status, audit messages, remote hosts, and web services). In Service Explorer, each time you send a test message to the back-end web service through the Enterprise Gateway, the message is displayed in the console. On the Status tab, you see a graph showing passed, blocked, and failed messages. The console can also display the entire execution of a transaction and enable you to see message payload and save it.

Note: Blocked messages are messages with malformed content.

Monitoring Using Service Monitor



ORACLE®

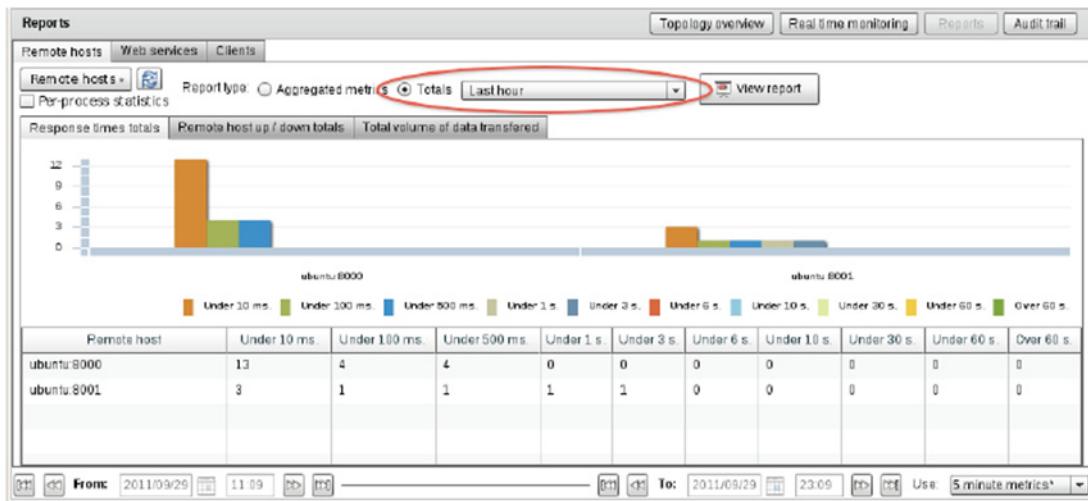
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Service Monitor enables you to view the traffic history of the Enterprise Gateway in reports so that you can analyze service usage.

To monitor the traffic between Enterprise Gateway instances and the various web services, clients, and remote hosts running throughout your network, you must add the Enterprise Gateway instance as a node on the “Topology overview” page. This page shows all nodes (Enterprise Gateway instances) that are sending monitored traffic to protected web services, clients, and remote hosts.

Each of these Enterprise Gateway nodes must already be configured to store message metrics in the same database from which Service Monitor is configured to read. This enables real-time monitoring, reports, and the audit trail to obtain the Enterprise Gateway metrics and logs that they display from this database.

Viewing Traffic Reports



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Service Monitor can then generate usage reports and charts based on the data stored in a centralized database by the Enterprise Gateway instances. The Reports screen is responsible for rendering these charts. The following configuration options apply to each of the three tabs available on this screen.

- **Aggregated metrics:** Shows metrics for a group of monitored objects (for example, all web services) across one or more nodes over a specific time period. This report displays the time line by using a graph. This enables you to view the total performance metrics for all web services, remote hosts, or clients.
- **Totals:** Shows all values for a specific monitored object (for example, a web service) over a specific time period. The report uses a bar chart to display the total value of a single metric for a monitored object over a specified time period. This enables you to compare the performance metrics of specific web services, remote hosts, or clients.
- **Time Window:** You must select the time window size that you want to generate reports when you use the button at the top of the screen.

Quiz

You need to configure a database for:

- a. Real-time monitoring
- b. Traffic Monitor
- c. Viewing reports in Service Monitor
- d. Tracing



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Roadmap

- Monitoring traffic
- Setting tracing levels
- Configuring log settings



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Logging Versus Tracing

- Logging deals with capturing information, which is used to:
 - Audit the activity of the gateway
 - Pinpoint essential issues
- You can log:
 - Specific transaction events
 - Payload (data flowing through the gateway)
- Log files can be digitally signed to guarantee integrity to entries.
- Tracing deals with capturing verbose information (usually to debug a problem).
- You can set trace at several levels: FATAL, INFO, DEBUG, DATA, and so on.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

OEG has a logging framework that records OEG events and messages and preserves those records for later examination. For example, the information can be used to audit the activity of the gateway and identify problems. Tracing captures more verbose information, which is useful for debugging a problem.

You can set trace levels to provide different amounts of trace information. The levels of tracing available include the following:

- FATAL
- INFO
- DEBUG
- DATA

DATA tracing is the most verbose level, while FATAL is the least verbose.

Accessing Trace and Logs Online

`http://localhost:8090`

The screenshot shows the Oracle Enterprise Gateway homepage. At the top, there's a red header bar with the text "ORACLE® Enterprise Gateway". Below it, the main content area has a white background. The title "Welcome to Oracle Enterprise Gateway" is centered at the top of the content area. Below the title is a bulleted list of links. The fourth item in the list, "[Audit trail log files](#)", is enclosed in a red rectangular box, indicating it is the target of the previous slide's instruction.

- [Traffic Monitor](#)
- [Real time monitoring console](#)
- [Service Manager](#)
- [Audit trail log files](#)
- [Trace files](#)
- [Diagnostic information](#)
- [Documentation](#)

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The traces and logs are available through the web interface. To access them, connect to `http://<OEG_Gateway_host>:<admin_port>/`.

Here, the admin port is 8090 by default (can be changed), or the port can be used over SSL.

Setting Enterprise Gateway Trace Levels

- Trace levels can be set to the following values (listed in order from least verbose to most verbose):
 1. FATAL
 2. ERROR
 3. INFO (default value)
 4. DEBUG
 5. DATA
- You can set the trace level in the following different ways:
 - Startup trace
 - System Settings trace
 - Interface-level trace



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The possible trace levels listed in order of least to most verbose output are (as shown in the slide):

1. FATAL
2. ERROR
3. INFO (default)
4. **DEBUG:** When you are troubleshooting, it can be useful to set the trace level to DEBUG for more verbose output. When running a trace at DEBUG level, the Enterprise Gateway outputs the status of every circuit and filter that it processes into the trace file.
5. **DATA:** The Enterprise Gateway writes the content of the messages that it receives and sends to the trace file.

You need to refresh the server configuration after changing the trace levels.

You can set the trace level in the following ways:

- **Startup trace:** When the Enterprise Gateway is starting up, it gets its trace level from the `tracelevel` attribute of the `SystemSettings` element in `/system/conf/enterprisegateway.xml`. You can set the trace level in this file if you need to diagnose bootup issues.

- **System Settings trace:** When the Enterprise Gateway has started, it reads its trace level from the System Settings for the Enterprise Gateway process. To set this trace level in Policy Studio, click the Settings button in the toolbar, select a Trace level from the drop-down list, and click OK.
- **Interface-level trace:** You can configure HTTP/HTTPS interfaces with a different trace level from System Settings. For example, the Management Services port (8090) has a default trace level set to ERROR to ensure that it is not too verbose at run time. To set an interface trace level in Policy Studio, right-click an HTTP/HTTPS interface port in the Processes tree and select Edit. Select a trace level from the drop-down list and click OK.

Viewing Enterprise Gateway Trace Files

- Each time the Enterprise Gateway starts up, it outputs (by default) a trace file to the `INSTALL_DIR\trace` directory.
- The trace file output takes the following format:

TraceLevel Timestamp [thread-id]
TraceMessage

Example:

```
INFO 13:58:29:687 [0804] rolling trace to file trace/Oracle  
Enterprise Gateway.trc started  
INFO 13:58:31:078 [0804] loaded netservice library  
INFO 13:58:31:406 [0804] attempting to connect to entity store  
at federated:file:///C:/enterprisegateway/conf/fed/configs.xml  
for process Enterprise Gateway ...
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows an extract from a default Enterprise Gateway trace file. The first line in the default trace file is described as follows:

- **Trace Level** INFO
- **Timestamp** 13:58:29:687 (hours:minutes:seconds:milliseconds)
- **Thread-id** 0804
- **Trace Message** rolling trace to file trace/Oracle Enterprise Gateway.trc started

Roadmap

- Monitoring traffic
- Setting tracing levels
- Configuring logging and audit trail settings



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Configuring Logging

- You can configure the following log settings:
 - What is logged
 - Where it is logged
 - Whether you want to log the request or response payload
- The log file output takes the following format:

[Log Level] [Timestamp] [Unique message ID]
[Failure Message] [Filter Type] [Filter name]

Example:

```
Failure 09.18.2009 17:47:16,536 Id-  
00000123cdd5fda2-000000001b71c12-17 'Filter  
failed' WSFilter Service Handler for  
'StockQuoteService'
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

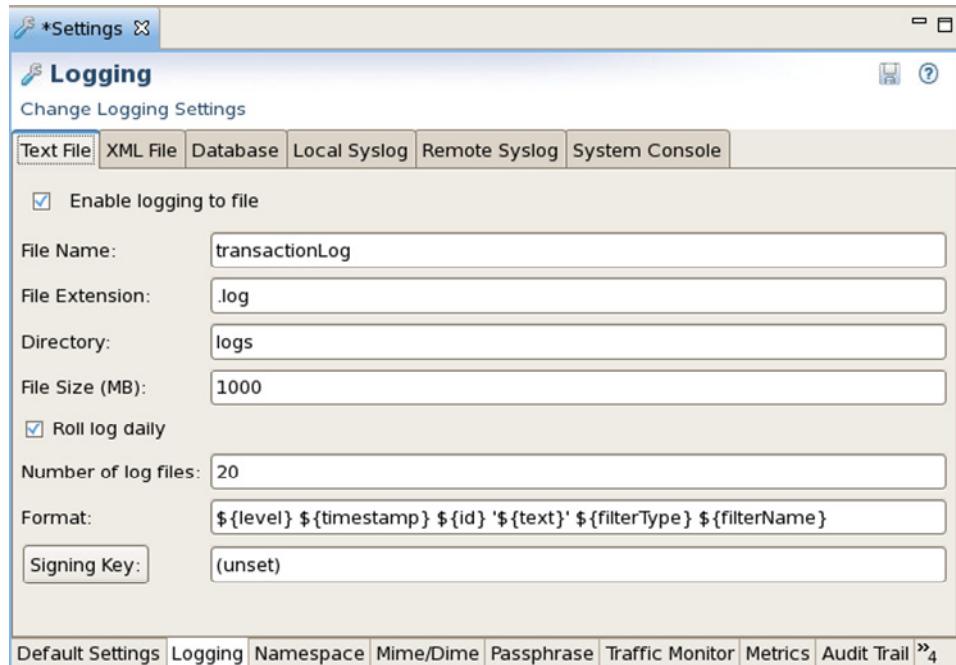
You can configure the Enterprise Gateway so that it logs information about all requests. Such information includes the request itself, the time of the request, where the request was routed, and the response that was returned to the client. The Enterprise Gateway can also digitally sign the logging information it sends to the log files and the database. This means that the logging information cannot be altered after it has been signed, thus enabling an irreversible audit trail to be created.

The Enterprise Gateway provides detailed logging for specific message filters (for example, the request, the time of the request, where the request was routed, and the response returned to the client).

You can configure logging to a number of different locations:

- Text file
- XML file
- Database
- Local syslog
- Remote syslog
- System console

Configuring Log Output

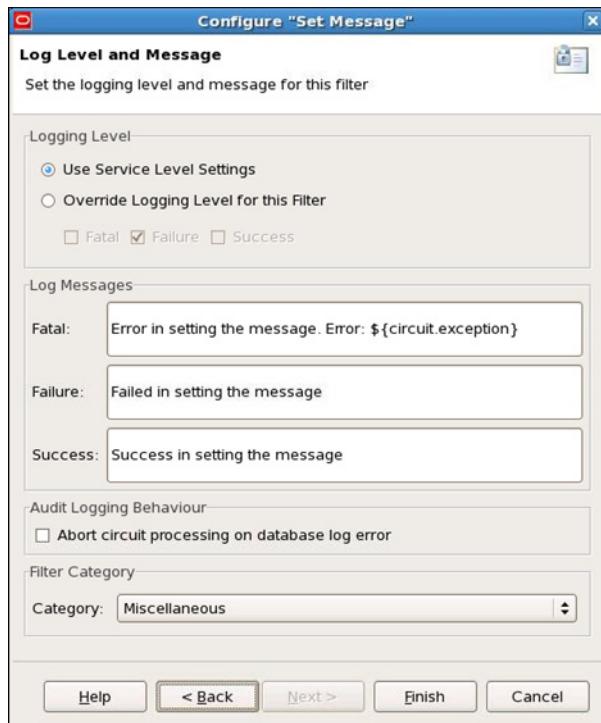


ORACLE

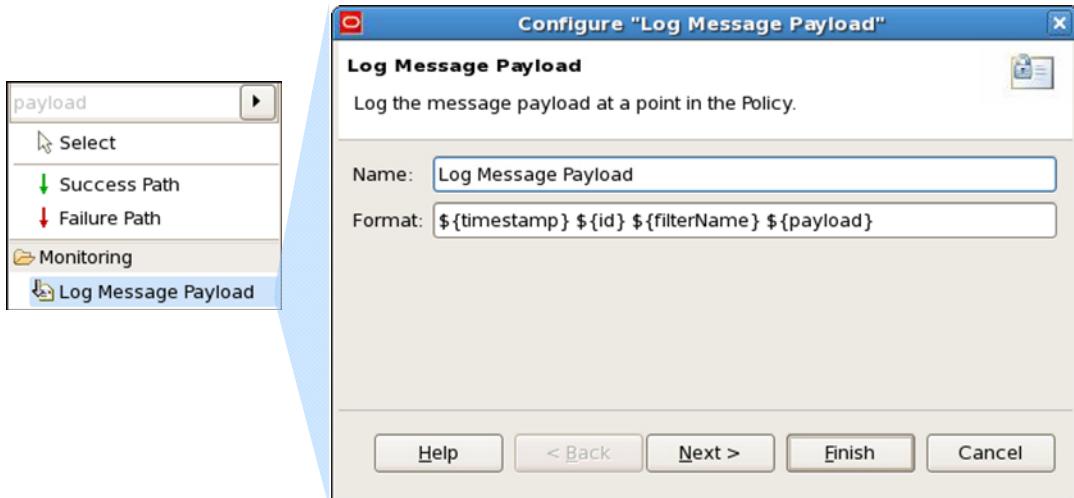
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This slide shows the log output configuration page in Policy Studio.

Configuring Log Level and Message



Log Payload Filter



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Log Message Payload filter is used to log the message payload at any point in the policy. The message payload also includes the HTTP headers and MIME/DIME attachments.

By placing the Log Message Payload filter at various key locations in the policy, a complete audit trail of the message can be achieved. For XML files, the payload of each message is enclosed in a CDATA section.

This is especially useful in cases where the message has been altered by the Gateway, for example by signing or encrypting the message, inserting security tokens, or by converting the message to another grammar by using XSLT.

Audit Trail

```
<AuditTrail>
<ProductDetails>
<Product name="Oracle Enterprise Gateway" version="6.3.0-2011-12-20" platform="Linux.x86_64"/>
<Date current="Wed, 01 Feb 2012 22:59:52 +0000" currentDateUTC="1328137192" TZ="UTC"/>
</ProductDetails>
<logEntry
    date="Wed, 01 Feb 2012 22:59:52 +0000"
    dateTIme="1328137192974"
    user="admin"
    type="Process"
    status="Pass"
    >
    <logText>Set the active configuration for &apos;&apos;Oracle Enterprise Gateway&apos;
</logEntry>

<logEntry
    date="Wed, 01 Feb 2012 22:59:53 +0000"
    dateTIme="1328137193046"
    user="admin"
    type="User"
    status="Pass"
    >
    <logText>Login by user &apos;admin&apos;.</logText>
</logEntry>

<logEntry
    date="Wed, 01 Feb 2012 22:59:53 +0000"
    dateTIme="1328137193504"
    user="admin"
    type="Process"
    status="Pass"
    >
    <logText>Get the active configuration fingerprint for &apos;&apos;Oracle Enterprise
</logEntry>
```

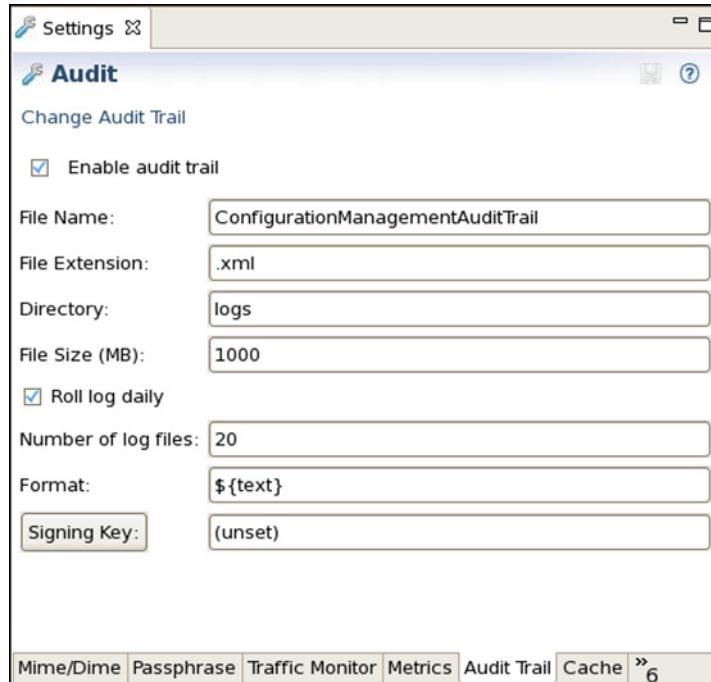


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Oracle Server Process (for example, Enterprise Gateway, Policy Center, or Service Monitor) generates an audit trail for each of the key actions that occurs in the Policy Studio on Configurations, Processes, and Users. For example, this includes when a user logs in, or when a process configuration is updated. All items are written to a file-based audit trail that is stored on the same machine as that on which the server is running. The audit trail rolls over on date change and also when a maximum file size is reached.

Logging and Audit trails keep a record of every transaction passing through the gateway.

Configuring Audit Trail



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can configure the following fields on the Audit Trail tab to force the server to write an audit trail to file:

- **Enable Audit Trail:** Select this check box to enable writing event data to the audit trail.
- **File Name:** Enter the name of the audit trail file in this field. When an audit trail file rolls over (either because the maximum file size has been reached, or because the date has changed), a suitable increment is appended to the file name. Defaults to ConfigurationManagementAuditTrail.
- **File Extension:** Enter the file extension for the audit trail file. Defaults to .xml.
- **Directory:** Enter the directory for the audit trail file. Defaults to logs.
- **File Size:** Specify the maximum size that an audit trail file is allowed to reach before it rolls over to a new file. Defaults to 1000 kilobytes.
- **Roll Log Daily:** Specify whether to roll over the log file at the start of each day. This is enabled by default.
- **Number of Files:** Specify the number of log files that are stored. The default number is 20.
- **Signing Key:** You can sign the audit trail to guarantee its integrity by using the private key of a system user. This user's key should be stored in Policy Studio's Certificate Store.

Quiz

When debugging a problem, you usually capture verbose information by setting:

- a. Log level to Fatal
- b. Log level to Data
- c. Trace level to Fatal
- d. Trace level to Data



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: d

Summary

In this lesson, you should have learned how to:

- Monitor traffic by using OEG monitoring tools
- Describe the difference between logging and tracing in OEG
- Configure log settings
- Set tracing levels
- View trace information



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 5 Overview: Monitoring, Logging, and Tracing

This practice covers the following topics:

- Enabling monitoring
- Monitoring using Traffic Monitor and the Real time monitoring console
- Viewing Reports by using Service Monitor
- Configuring logging and trace



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Managing Configurations

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe OEG configuration structure
- Manage a deployed configuration
- Manage configuration versions
- Import and export configuration data



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Configuration Management

- Policy Studio:
 - Edit configurations.
 - Create different configuration versions.
 - Import and export configurations.
- Policy Studio and Policy Center:
 - Centrally manage the policies.
 - Deploy configuration to multiple Enterprise Gateway instances.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

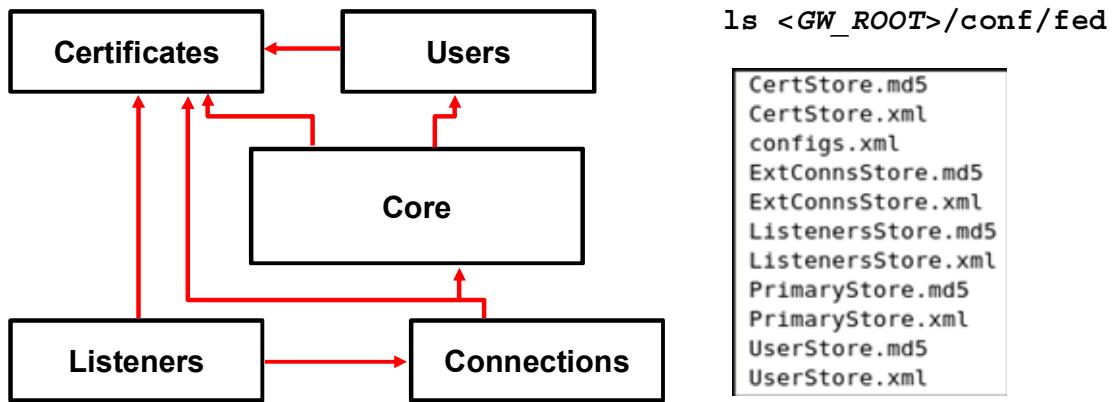
You can use Policy Studio to edit the configuration of currently running Enterprise Gateway server instances, and specify different configuration versions. When connected to a Policy Center server, you can deploy configurations to multiple processes managed by Policy Studio. Managed processes can be Enterprise Gateway, Policy Center, and Service Monitor server instances.

In this way, Policy Studio and Policy Center enable administrators to centrally manage the policies that are enforced at all nodes. In addition, Policy Studio enables administrators to compare and merge differences between versions of the same policy. Policies can be merged, and deployed to any running process that is managed by Policy Studio.

One of the scenarios in which this centralized management capability is most useful is in transitioning from a staging environment to a production environment. For example, policies can be developed and tested on the staging environment and, when ready, they can be deployed to all processes deployed in the production environment.

Configuration Structure

- The configuration is stored in `<GW_ROOT>/conf/fed`.
- There is one XML file per configuration section.
- Each configuration section can be edited or versioned.



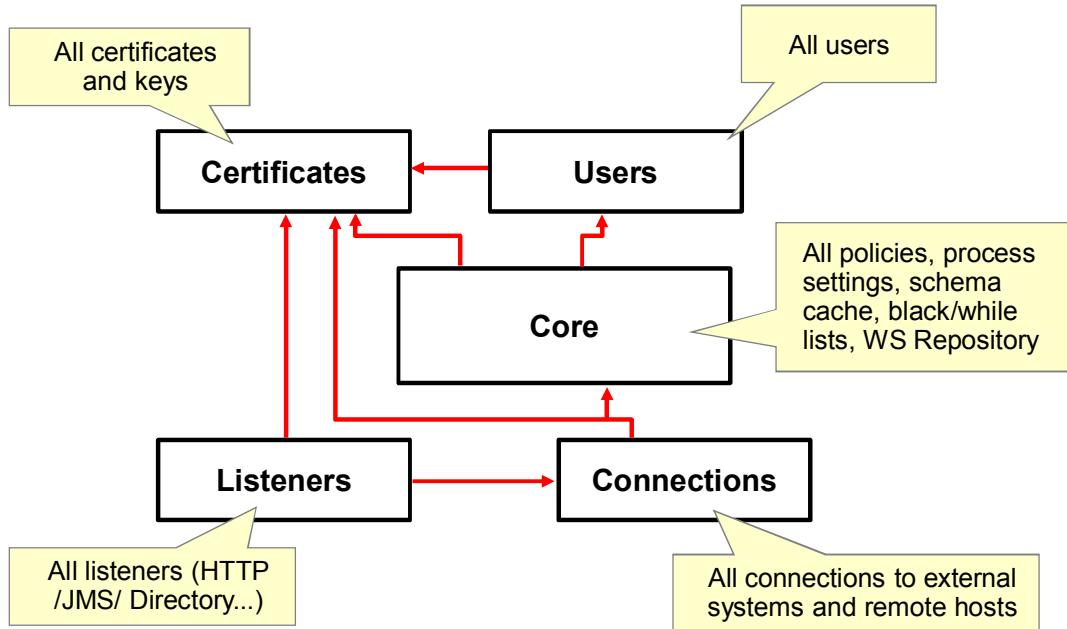
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The configuration of the Enterprise Gateway instance is stored in the `<GW_ROOT>/conf/fed` directory. It consists of several sections: core (PrimaryStore), certificates, users, listeners, and external connections. Each configuration section is a stand-alone XML file and can be individually edited or versioned.

Administrators can use Policy Studio to specify different configuration versions. In Policy Studio, the Enterprise Gateway allows you to edit the configuration of currently running Enterprise Gateway server instances, or processes. You can then update the downloaded configuration and commit it to the server, where it can be reloaded later.

Storing Configuration Items



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The diagram shows the configuration items that are stored in each section.

Loading Configurations

- Policy Studio:
 - Loads the configuration from the server at startup
 - Works on a local configuration
- Service Manager:
 - Loads the configuration from its local copy (initially created from the server configuration)
 - Keeps a local copy of the configuration across working sessions



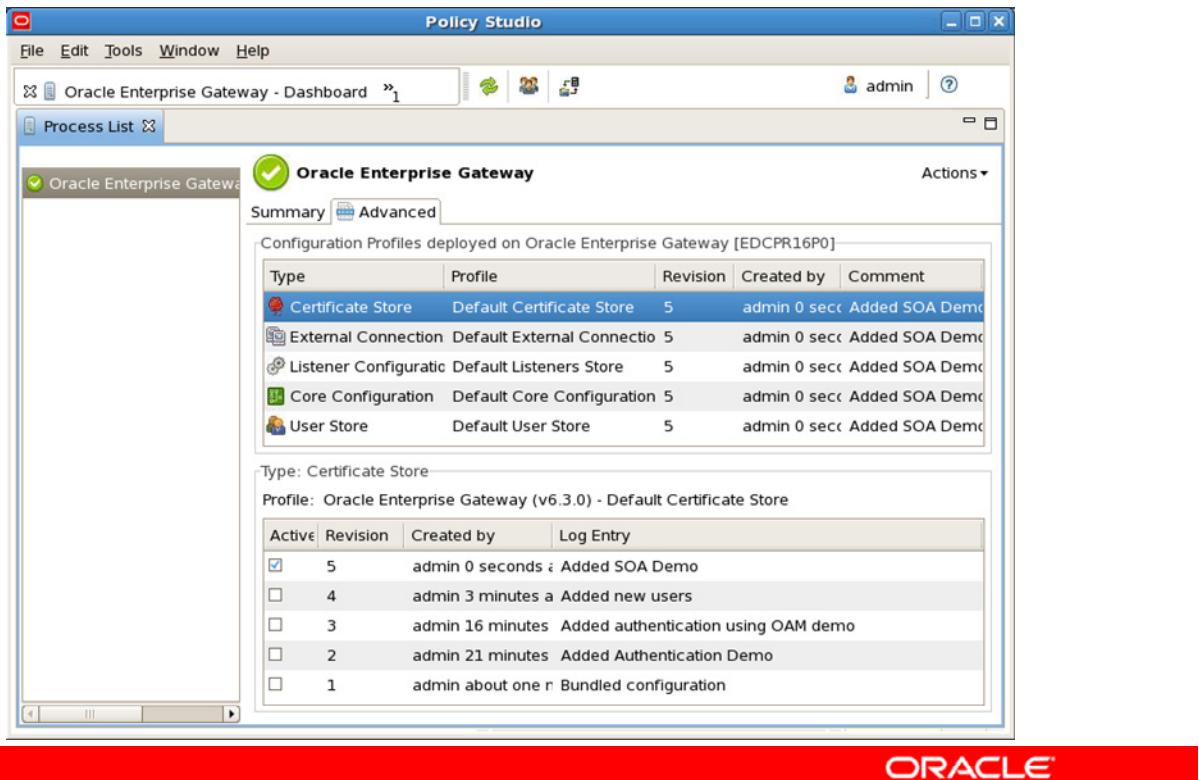
- Watch for this message at the bottom of the window.
- You can revert to the server configuration by clicking the Discard button.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- Policy Studio loads the configuration from the server at startup and then works on a local configuration. The local configuration is lost if you close the editor without deploying your changes.
- Service Manager loads the configuration from its local copy (which is initially created from the server configuration) and keeps the local copy of the configuration across working sessions.

Viewing Deployed Configuration



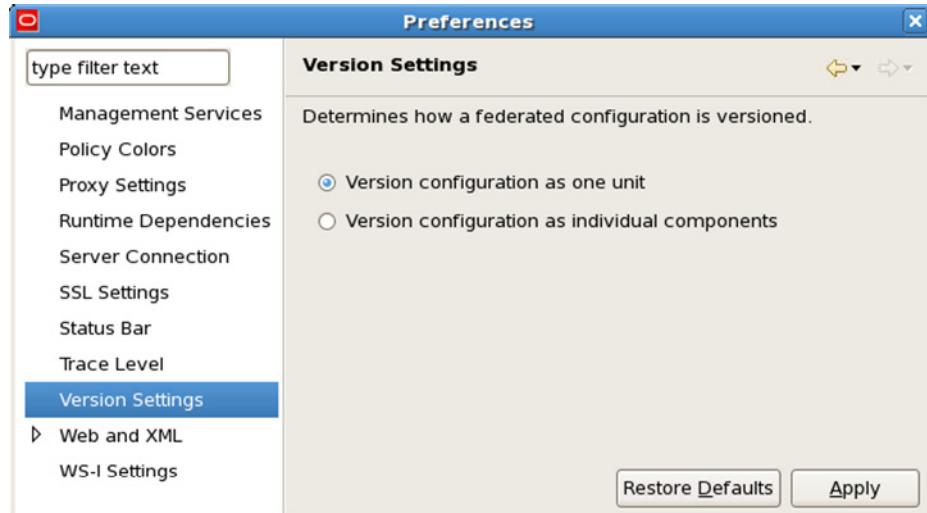
ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Oracle Enterprise Gateway Dashboard enables you to manage the component configuration stores that are currently deployed on a selected process. These include the configuration stores for core policies and services, external connections, users, and certificates.

For example, you can select a process and view the versions of the component configuration stores that are currently running on that process. You can also change the currently deployed versions, and deploy your updates to the Enterprise Gateway server, or apply tags to a specific component configuration store.

Version Settings



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Version Settings in Policy Studio Preferences enable you to specify whether the process configuration is versioned as a single unit or as individual components. By default, the entire process configuration is versioned as a single unit.

Versioning Process Configuration

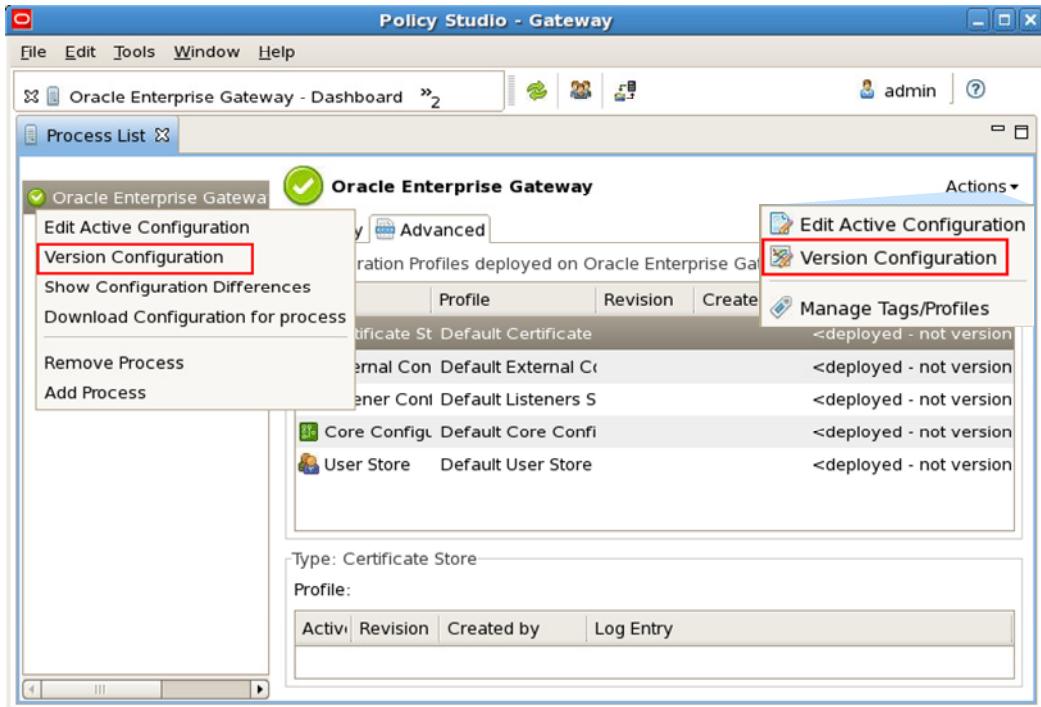
- Versioning requires that you deploy your configuration first.
- You can version the process configuration in:
 - Oracle Enterprise Gateway Dashboard
 - Active Server Configuration



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A process configuration must be deployed before you can apply a version and a comment. You can version the deployed configuration by using Oracle Enterprise Gateway Dashboard. When editing an active process configuration, you can also version deployed updates directly.

Versioning Configuration in Oracle Enterprise Gateway Dashboard



ORACLE

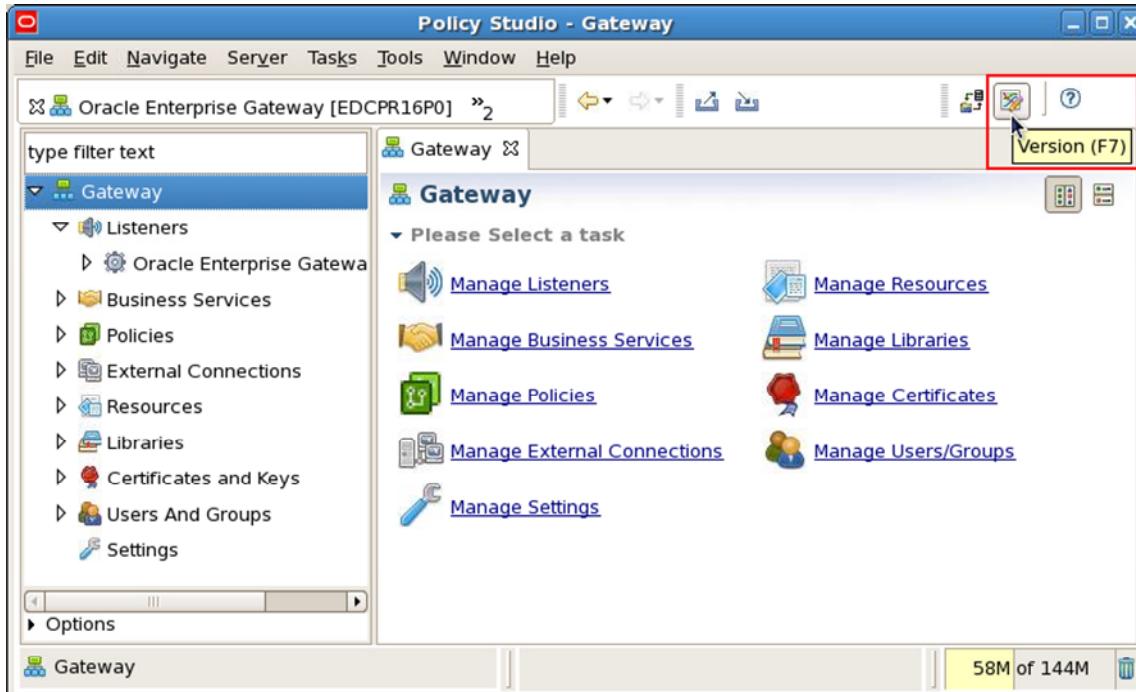
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To version the configuration of a selected process in Oracle Enterprise Gateway Dashboard, you can click the Action menu, and select the Version Configuration link. Alternatively, right-click the process in the Process List, and select Version Configuration.

Enter a Comment to version the configuration in the Versioning dialog box.

If you selected to version the process configuration as individual components in Policy Studio Preferences, select whether to apply this comment to all modified components (the default), or to apply individual comments to selected modified components.

Versioning Configuration in Active Server Configuration



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When editing an active process configuration, you can apply a version and a comment to a deployed process configuration using the Server > Version menu option, or the Version button in the toolbar. Alternatively, you can press F7.

Quiz

You can version a process configuration only as one unit.

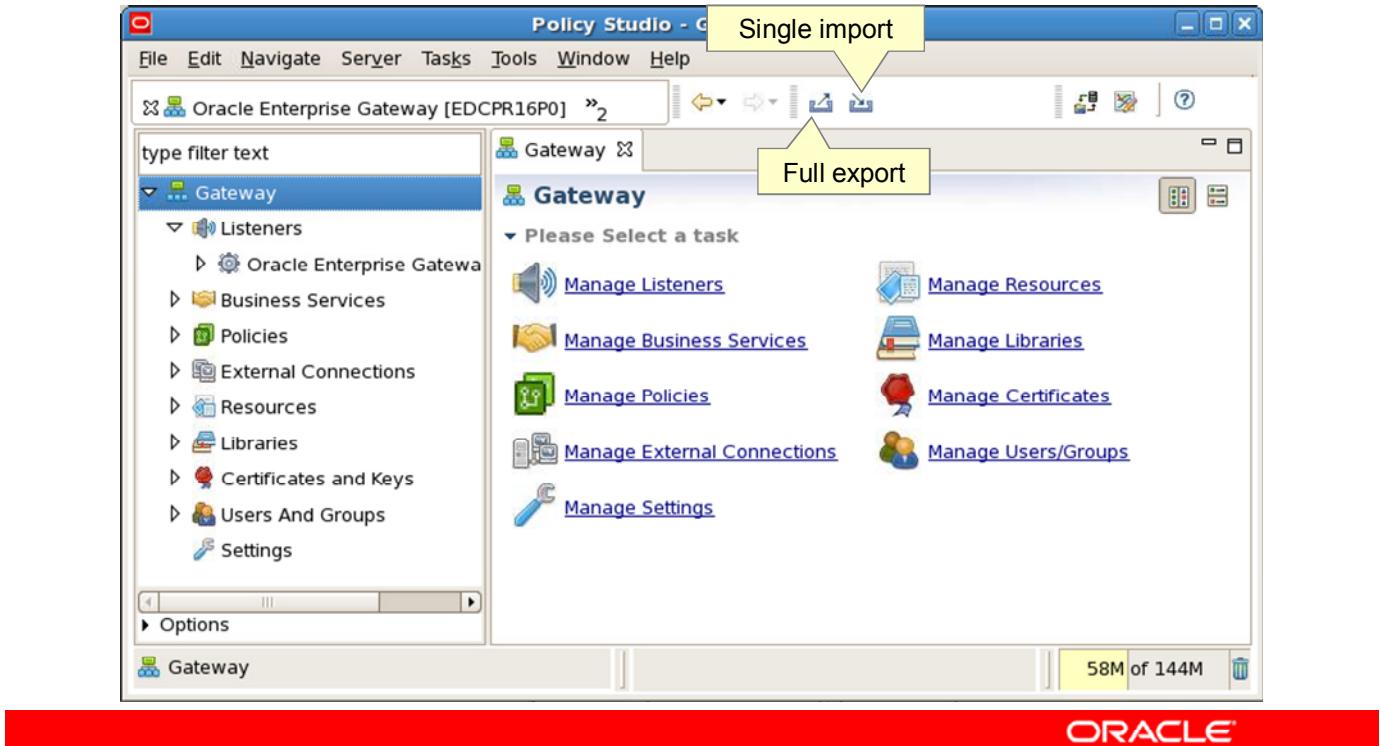
- a. True
- b. False



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Importing and Exporting Configurations



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE

You can export (or import) configuration data out of (or into) your Enterprise Gateway configuration (for example, policies, certificates, and users). The configuration is exported to an XML file, which you can then import into a different Enterprise Gateway configuration. This is useful if you have configured the Enterprise Gateway in a testing environment and want to move this configuration to a live production environment. By exporting configuration data from a testing environment and importing it into a production environment, you can effectively migrate your Enterprise Gateway configuration. This is the recommended way to export Enterprise Gateway configuration data, and it enables you to manage references between configuration components.

Exportable Configuration Items

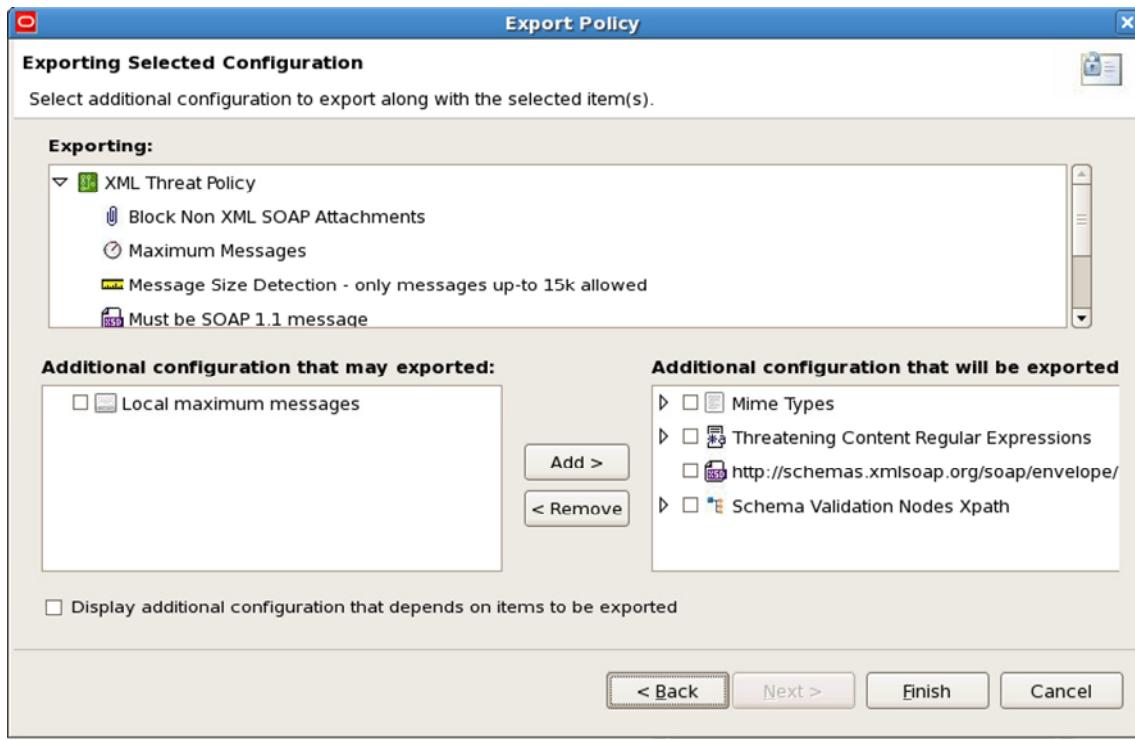
- Policies
- Policy Containers
- Schemas
- Registered Web Services
- Remote Host
- Relative Path
- External Connections
- Local Repository
- Certificates (public/private)



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can export an Enterprise Gateway configuration item by right-clicking a node in the Policy Studio tree. This slide lists the elements that can be exported from the gateway. In addition, you can export configuration items that are associated with the selected elements. For example, this includes referenced policies, MIME types, regular expressions, schemas, and remote hosts.

Exporting Configuration Data

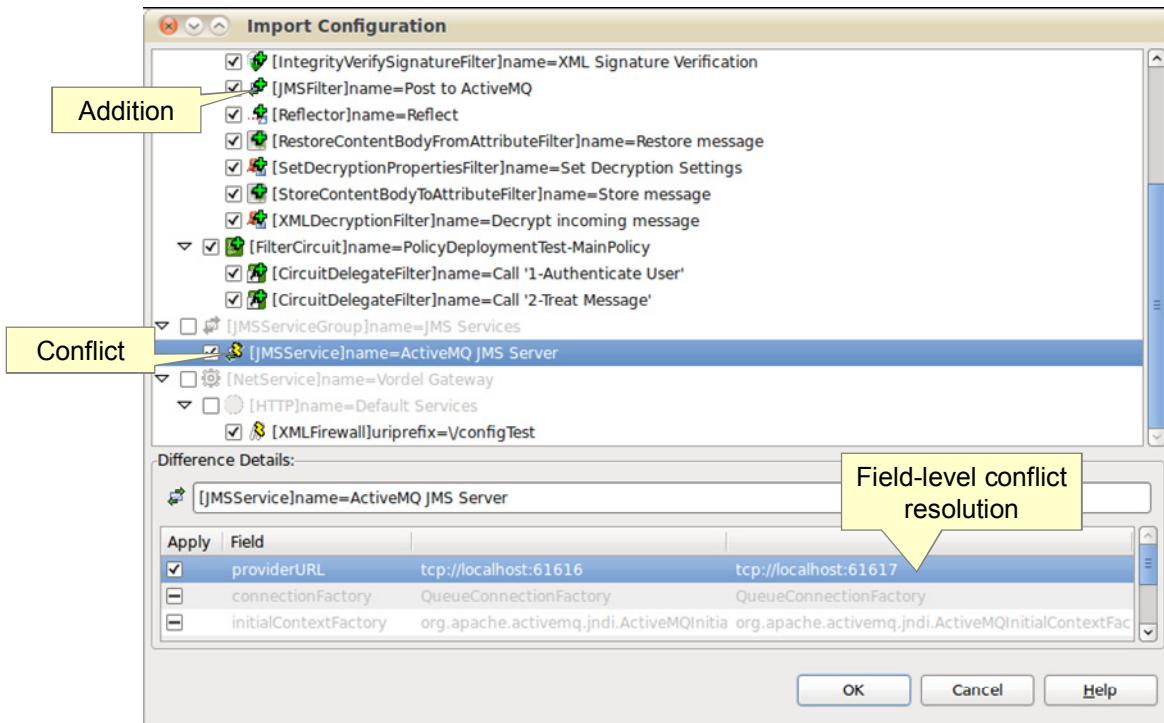


ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can export Enterprise Gateway configuration data by right-clicking a Policy Studio tree node (for example, Policy or Policy Container) and selecting the relevant Export menu option. The configuration is exported to an XML file, which you can then import into a different Enterprise Gateway configuration. In the example in the slide, a policy is exported. You can selectively export referenced, dependent, and supporting elements.

Importing Policies



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When you import configuration items (for example, a policy), the Import Configuration dialog box displays the differences between the existing stored configuration data (destination) and the configuration data to be imported (source). Differences are displayed in the tree as follows:

- Addition exists in the source configuration being imported but not in the destination configuration:** Displayed as a green plus (+) symbol
- Deletion exists in the destination configuration but not in the source configuration being imported:** Displayed as a red minus (-) symbol
- Conflict exists in both configurations but is not the same:** Displayed as a yellow warning symbol

If you select a particular node in the Import Configuration tree, the Difference Details panel at the bottom of the screen shows details for this configuration entity (for example, added or removed fields). In the case of conflicts, changed fields are highlighted. Some configuration entities also contain references to other entities. In this case, a symbol is displayed for the field in the Difference Details panel. If you double-click a row with a symbol, you can drill down to view additional Difference Details dialog boxes for those entities.

Deploying the Gateway in Multiple Environments

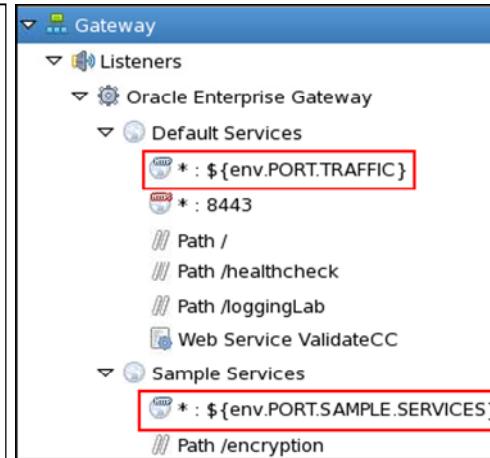
Environment variables:

- Enable you to define and override environment-sensitive message attributes.
- Are defined in the `<gateway_dir>/conf/envSettings.props file`

```
# the default port which the Gateway
listens on for HTTP traffic
env.PORT.TRAFFIC=8080

# the default port which the Gateway
listens on for sample messages
env.PORT.SAMPLE.SERVICES=8081

# the default port which the Gateway
listens on for management/configuration
HTTP traffic
env.PORT.MANAGEMENT=8090
```



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can specify configuration values in the Enterprise Gateway on a per-environment basis using environment variables in the `envSettings.props` file. For example, you can specify the port on which the Enterprise Gateway listens for HTTP traffic with different values depending on the environment in which the Enterprise Gateway is deployed.

The environment variable settings in the `envSettings.props` file are external to the Enterprise Gateway core configuration. The Enterprise Gateway run-time settings are determined by a combination of external environment variable settings and core configuration policies. This mechanism provides a simple and powerful approach for configuring the Enterprise Gateway to work across multiple environments.

The `envSettings.props` file is located in the `conf` directory of your Enterprise Gateway installation, and is read each time the Enterprise Gateway server starts up. The environment variable values specified in the `envSettings.props` file are displayed as environment variable properties in the Policy Studio.

Quiz

The process configuration is exported to:

- a. An XML file
- b. A database
- c. A plain text file
- d. The System Console



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe OEG configuration structure
- Manage a deployed configuration
- Manage configuration versions
- Import and export configuration data



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 6 Overview: Managing Configurations

This practice covers the following topics:

- Versioning process configuration
- Exporting the configuration data



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



Fault Handling

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Change default fault information that is passed back by OEG
- Register a fault handler
- Use OEG's trace to see why messages are blocked



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Abort Versus Failure

- Aborts are fatal errors.
 - Aborts occur rarely in OEG.
 - Aborts are handled by fault handlers.
 - You can force an abort by using an Abort filter.
- Failures are recoverable errors.
 - Failures are handled in “Failure Paths” (red arrows).
 - Failures are used by OEG in the majority of cases.
 - You can force a failure by using the False filter.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

OEG product interruptions are inevitable. An interruption can be one of two common types: abort and failure.

Aborts are fatal errors and, therefore, occur rarely. The most common case is invalid XML that cannot be parsed by OEG. The Abort filter can be used to force a policy to throw an exception. It can be used to test the behavior of the policy when an exception occurs.

Failures are less serious errors, and they are recoverable. They are used in the majority of cases and handled in the “Failure Paths” in a policy. The False filter can be used to force a path in the policy to return false.

Handling Failures

Treating failures:

- Using the default SOAP handler – Basic fault message
- Using the SOAP Fault filter – More complicated fault message
- Using a policy as a fault handler – Custom fault message



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

If an error occurs while a web service is processing a message, the Enterprise Gateway returns a very basic SOAP Fault to the client. It is possible to override the default SOAP message in case of execution errors. To do this, you need to provide what is called a fault handler. A fault handler can be a SOAP Fault filter, or a policy to be executed, which allows you to virtually treat errors the way you want, for example sending alerts, logging the event into a database, or returning a custom message to the client.

SOAP Faults

- During the processing of a request, if an error occurs, a web service generates a SOAP fault.
- SOAP faults may be generated based on business logic errors or unexpected conditions
- The SOAP specification defines the standard structure of a fault within a SOAP message:
 - Location within the SOAP envelope
 - Contained information



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

If an error occurs while a web service is processing a message, the client may have to be notified. In such cases, because a client application may be written on a different platform or language, there must exist a standard, platform-independent mechanism for describing the error. The SOAP specification defines SOAP faults for such situations. The specification describes its format within the SOAP envelope as well as its contents. The end result is that when an error occurs in a web service, no matter what language the service is written in, a SOAP message containing a SOAP fault will be used to communicate the error to the client.

SOAP faults are generated by receivers of a message. The receiver needs to send a SOAP fault back to the sender only if the Request/Response messaging mode is used. In One-Way mode, the receiver should generate a fault and may store it somewhere, but it must not attempt to transmit it to the sender.

SOAP Fault: Example

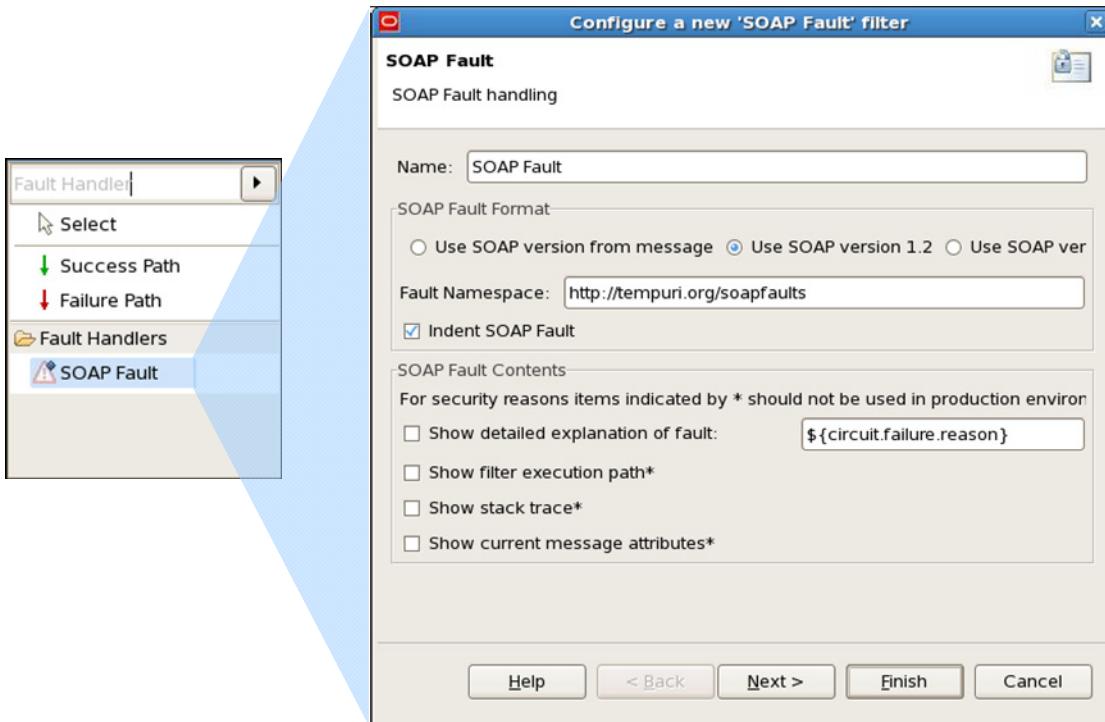
```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>Receiver</env:Value>
        <env:Subcode>
          <env:Value>policy failed</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Detail xmlns:oraclefault="http://www.oracle.com/soapfaults"
                  oraclefault:type="exception" type="exception"/>
    </env:Fault>
  </env:Body>
</env:Envelope>
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The SOAP `<Fault>` element is used to carry error and status information within a SOAP message. A SOAP message that contains a `Fault` element in the body is called a fault message. When a fault message is generated, the body of the SOAP message must contain only a single fault element and nothing else.

SOAP Fault Filter



ORACLE®

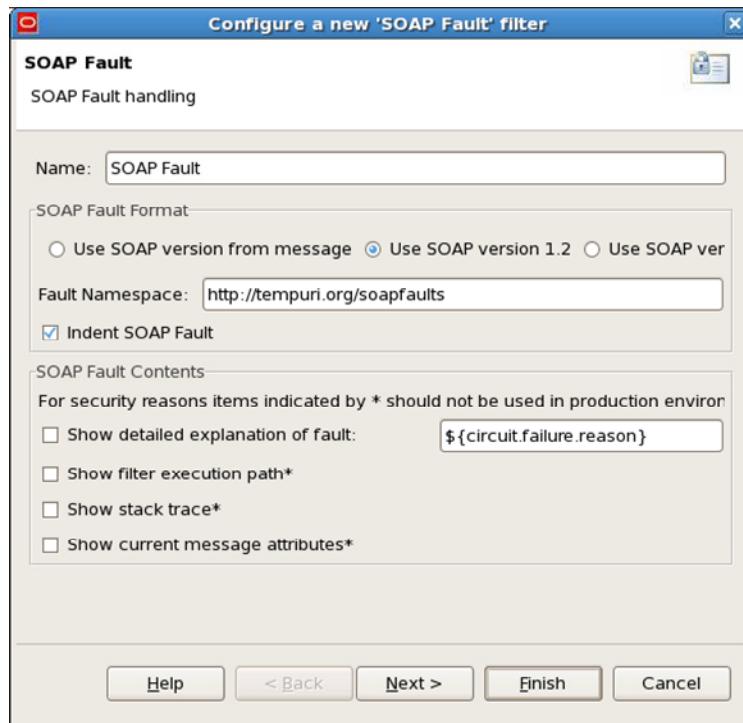
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In cases where a typical message fails, a SOAP Fault can be used to convey error information to the SOAP client. By default, the Enterprise Gateway returns a very basic SOAP Fault to the client when a message filter fails. The SOAP Fault filter can be added to a policy to return more complicated error information to the client.

For security reasons, it is a good practice to return as little information as possible to the client. However, for diagnostic reasons, it is useful to return as much information to the client as possible. Using the SOAP Fault processor, you have the flexibility to configure just how much information to return to clients depending on their individual requirements.

For security reasons, you should not return too much detailed information in an untrusted environment.

Configuring a SOAP Fault Filter



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

There are two sections that you can configure for a SOAP Fault filter: SOAP Fault Format and SOAP Fault Contents.

SOAP Fault Format

The following configuration options are available in this section:

- **SOAP version:** It is possible to send a SOAP Fault 1.1 or SOAP Fault 1.2 response to the client.
- **Fault Namespace:** All fault elements should be namespace qualified. A default namespace for SOAP Faults is prepopulated. You can use it or enter a new one if necessary.
- **Indent SOAP Fault:** If this option is selected, an XSL stylesheet is applied to the SOAP Fault to indent nested XML elements. The indented SOAP Fault is returned to the client.

SOAP Fault Contents

- **Show detailed explanation of fault:** If this option is selected, a detailed explanation of the SOAP Fault is returned in the fault message.
- **Show filter execution path:** If this option is selected, the Enterprise Gateway returns a SOAP Fault containing the list of filters run on the message before the error occurred. For each filter listed in the SOAP Fault, the status (“pass” or “fail”) is given.
- **Show stack trace:** If this option is selected, the Enterprise Gateway returns the Java stack trace for the error to the client. This option should be enabled only under instructions from the Oracle Support team.
- **Show current message attributes:** If this option is selected, the message attributes that are present at the time the SOAP Fault was generated are returned to the client.

Quiz

SOAP Fault is used to convey error information to the:

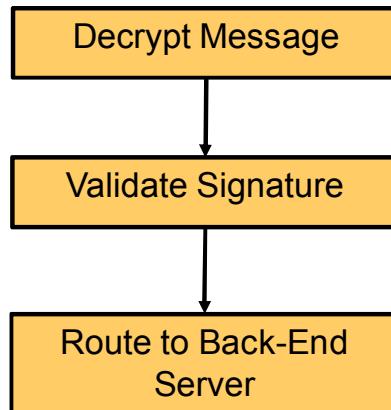
- a. System administrator
- b. SOAP message receiver
- c. SOAP message sender
- d. None of the above



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Example of Fault Handling



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You now see an example of how to treat failures.

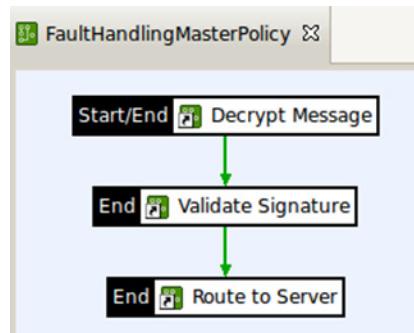
Suppose that you receive a message with an encrypted signature. The policy will:

- Need to decrypt and then validate it
- Route requests to a back-end server

Each step is implemented by calling another policy.

A custom error message must be returned if problems occur (decrypt failed, validation failed, and so on).

Default Fault Handler



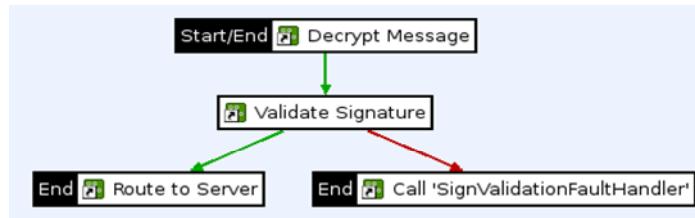
```
ERROR 11:44:12:779 [4471b70] Filter 'Validate Signature' Status:  
FAILEDERROR 11:44:12:779 [4471b70] Policy 'SignatureValidation' {  
ERROR 11:44:12:779 [4471b70] Filter 'Check if WS-Sec header exists'  
Status: FAILEDERROR 11:44:12:779 [4471b70] Filter 'Check Signature  
in SOAP Header' Status: FAILEDERROR 11:44:12:779 [4471b70]} ERROR  
11:44:12:779 [4471b70] } ERROR 11:44:12:779 [4471b70] Validate  
Signature filter failedDEBUG 11:44:12:779 [4471b70] run filter  
[com.vordel.circuit.soap.SOAPFaultProcessor@162a121] {DEBUG  
11:44:12:779 [4471b70] using fault creator: class  
com.vordel.circuit.soap.SOAP12FaultDocumentCreatorDEBUG 11:44:12:780  
[4471b70] } = 1, in 1 milliseconds
```

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In the example, all filters are marked with End, which means that the failure path is not handled (there are no red arrows). If the Validate Signature filter fails, the whole policy is marked as failed. Because you have not defined anything to handle this error, the default (system) fault handler is used, which returns some basic SOAP Fault.

Custom Handling for Failures



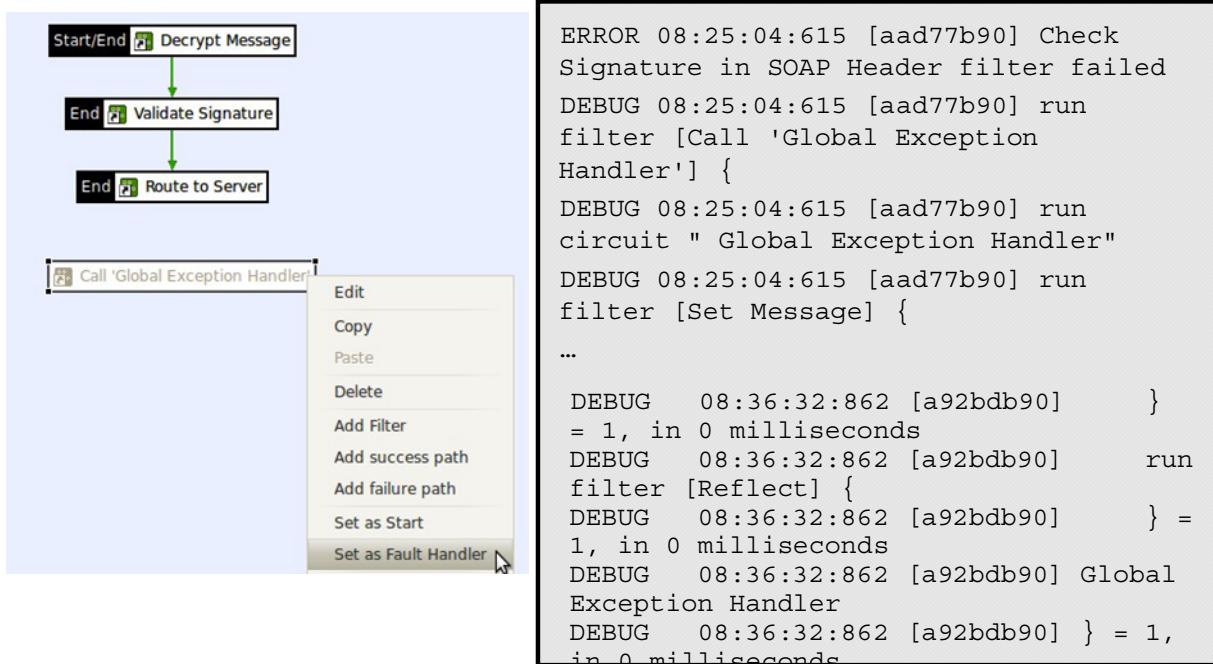
```
ERROR 08:36:32:861 [a92bdb90] The message [Id-000001249f397329-00000000018ff4f6-35] logged Failure ...
DEBUG 08:36:32:862 [a92bdb90] run filter [Call 'SignValidationFaultHandler'] {
DEBUG 08:36:32:862 [a92bdb90]     run circuit
"SignValidationFaultHandler"
DEBUG 08:36:32:862 [a92bdb90]     run filter [Set Message] {
...
DEBUG 08:36:32:862 [a92bdb90]         } = 1, in 0 milliseconds
DEBUG 08:36:32:862 [a92bdb90]         run filter [Reflect] {
DEBUG 08:36:32:862 [a92bdb90]             } = 1, in 0 milliseconds
DEBUG 08:36:32:862 [a92bdb90]             SignValidationFaultHandler
DEBUG 08:36:32:862 [a92bdb90]         } = 1, in 0 milliseconds
```

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To treat the specific Validate Signature policy failures, you need to add a failure path to the Master policy. The End marker for the Validate Signature filter is no longer present.

Overriding the Default Fault Handler

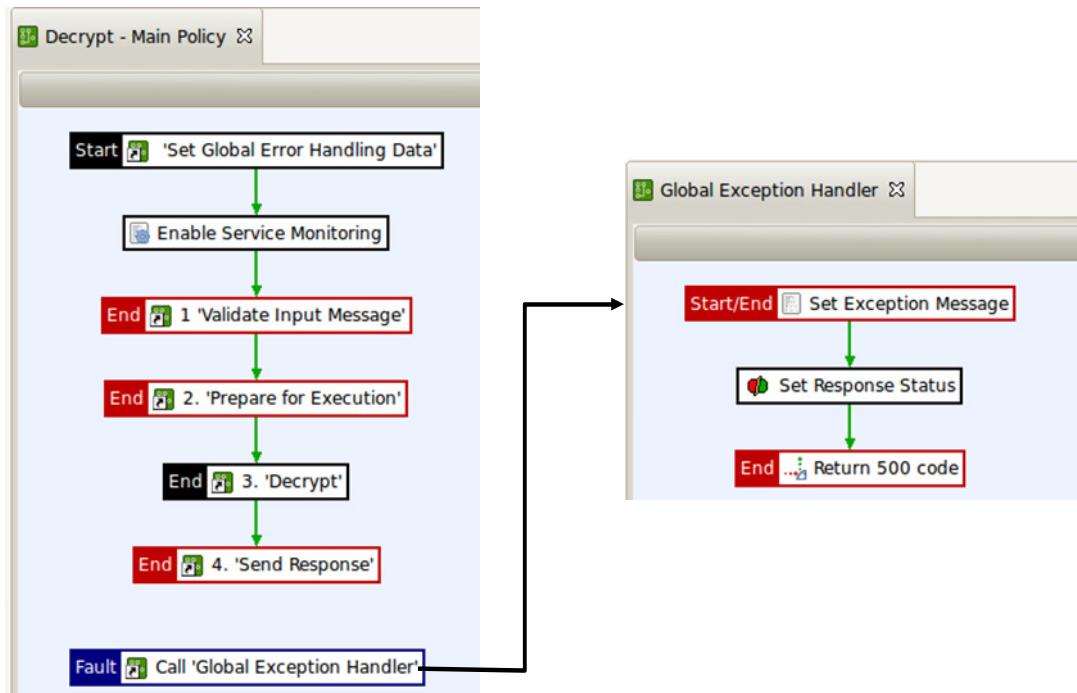


ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To override the default SOAP handler, you must define a fault handler in the root policy. The fault handler can be a SOAP Fault filter or a call to another policy. In this example, use another policy called `Global Exception Handler` to be the fault handler.

Custom Fault Handling by Using a Policy



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

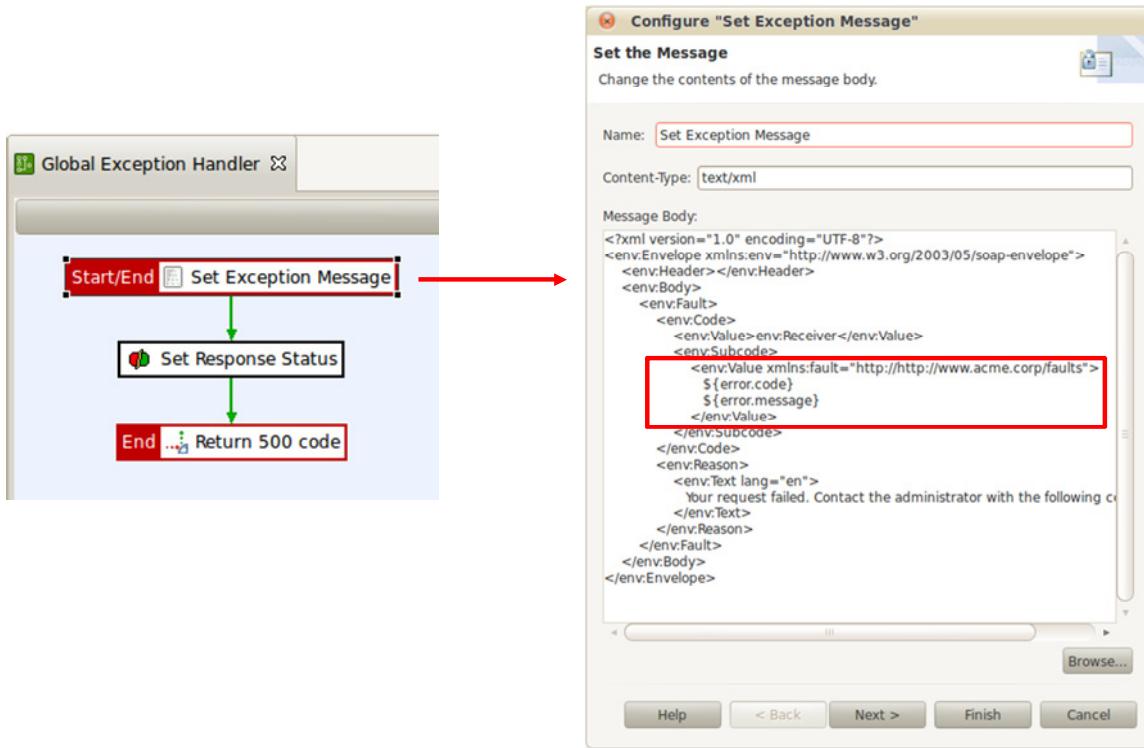
It is possible to create customized SOAP Faults by using the Set Message filter. The Set Message filter can change the content of the message body to any arbitrary content. When an exception occurs in a policy, it is possible to use this filter to customize the body of the SOAP fault. You can use any policy to handle a fault, through a policy shortcut.

To generate customized SOAP faults and return them to the client:

1. Create a shortcut to the fault policy.
2. Create the top-level policy.
3. Create the fault policy.

When an exception is thrown in a policy, it is handled by the designated fault handler, if one is present. In the “Main Circuit,” you have set a Policy Shortcut filter to be the fault handler. This filter delegates to the “Fault Circuit,” meaning that when an exception occurs, “Main Circuit” invokes (or delegates to) the “Fault Circuit.”

Example of Custom Fault Handling: Global Handler



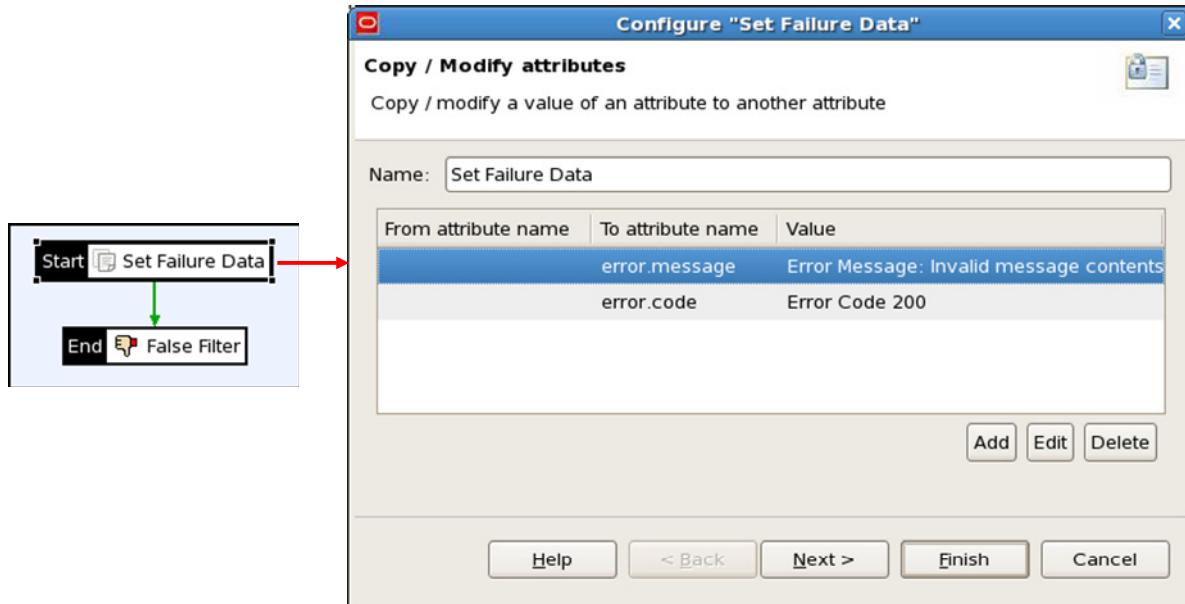
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE

This and next slides show an example of custom fault handling using the Global Handler and the Specific Handler together. The Global Handler has these three filters:

- **Set the Message:** Set what exception message to be thrown out, with custom error code and message. `${id}` is returned in the message to indicate to the consumer which transaction failed.
- **Set Response Status:** Sets monitoring status to fail
- **Reflect Message:** Sets custom error

Example of Custom Fault Handling: Specific Handler



ORACLE®

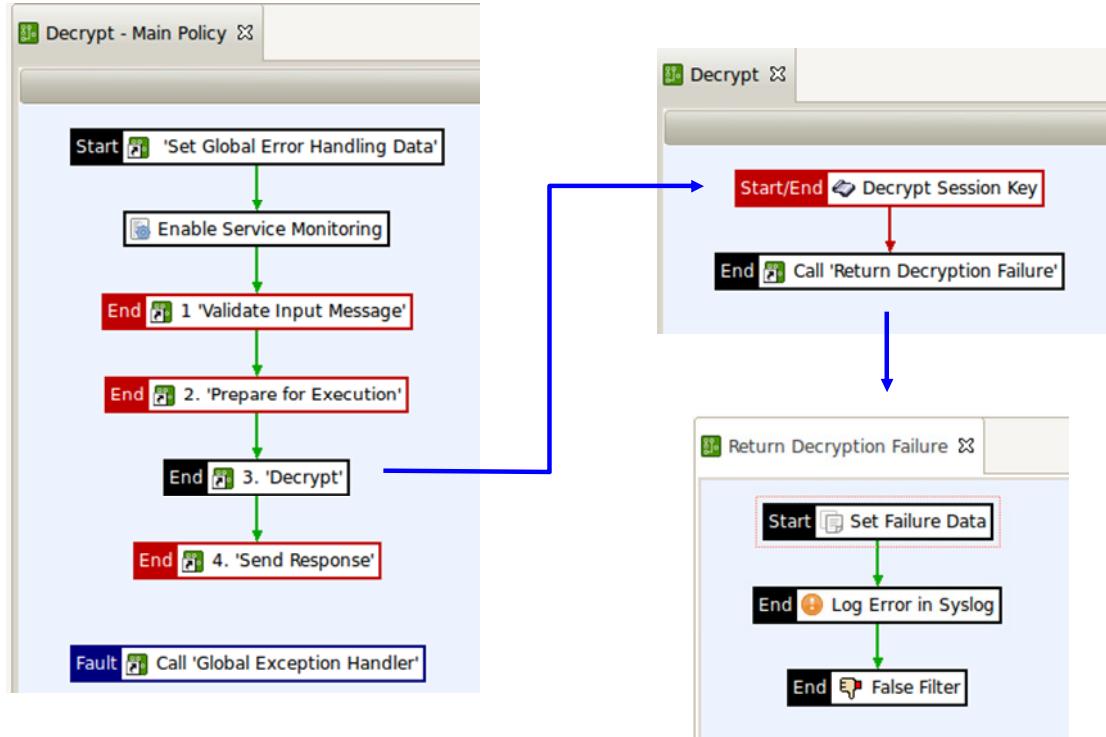
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Specific Handler has two filters:

- **Copy / Modify attributes:** Sets the value of error.code and error.message
- **False filter:** Returns false to propagate exception up the stack

Global Exception Handler is then invoked.

Common Use Case: Propagating Exceptions



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This slide shows a common use case of a fault handler.

By connecting anything to a failure transition, the exception is considered to be handled. A result with the value “true” is returned to the calling policy, and execution of the main policy therefore continues. To avoid this, always make sure to return false (using the False Filter) so that the exception is sent up the stack.

Monitoring Impact



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can monitor the impact of the fault processor. If an error is handled properly, the message is considered to be successful.

If you want to mark a message as failed, you can use the Set Response Status filter.

Handling Aborts



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Abort filter is used to simulate an abort if the signature cannot be found in the SOAP header.

If the Validate Signature policy aborts, the error is caught by the nearest fault handler (in this case, the Master policy fault handler).

Quiz

Which of the following options are used to override the default SOAP handler of a policy?

- a. Set Message filter
- b. SOAP Fault filter
- c. False filter
- d. A custom fault handling policy



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b, d

Summary

In this lesson, you should have learned how to:

- Change default fault information that is passed back by OEG
- Register a fault handler
- Leverage OEG's trace to see why messages are blocked



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 7 Overview: Using Customized Fault Handler

This practice covers the following topics:

- Testing the service by using the default fault handler
- Importing and viewing the fault handling policy
- Adding Global Fault Handler



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

8

Blocking XML Threats

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the different types of XML threats
- Identify the filters that block specific threats



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

XML Concepts

- Namespaces: To interpret the tag in the proper context
- XML Schema (XSD): To validate the integrity of XML documents
- XPath: A language for addressing (finding, retrieving, or manipulating) values in XML sources
- XML Stylesheet Language for Transformations (XSLT): A language, itself written in XML, used for the transformation of XML documents



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The primary XML concepts and technologies include:

- **Namespaces:** When duplicate element names exist in a document, the result is a namespace collision. The W3C namespace specification defines mechanisms for qualifying the names so as to eliminate ambiguity—that is, provide unique names for all elements and attributes. XML namespace provides a qualified name for an XML element or attribute.
A namespace is an attribute=value pair just below the prolog in an XML document, and uses the syntax: `namespace : attribute=URI`
Here is an example of a namespace declaration:
`xmlns:java=http://www.javasoft.com/2006/software`
- **XML Schema:** An XML Schema is an XML document that defines the syntax of XML documents governed by the schema. The element in an XML instance document has to comply with the XSD definition to be considered valid by XML processors. Data types in schemas have two general forms: simple and complex. Simple is a terminal element; complex contains a sequence of child elements.

- **XPath:** XPath is a language designed to address specific parts of an XML document. XPath defines a syntax to pinpoint just a portion of an XML document. This is very useful if just part of an XML document is to be encrypted, or is to be digitally signed. Using XPath, one can construct an expression (location path) that accesses a node or set of nodes. XPath is a major element in W3C's XSLT standard. Refer to <http://www.w3.org/TR/xpath> for the specification.
- **XSLT:** XML documents represent data, organized in some predefined, structured way. Sometimes, data in XML documents needs to be transformed before it can be sent or processed. Source and target documents are generally based on different XSD documents. XSLT is a language that defines the XML elements and attributes to produce and the values for these to write, usually queried from the source document. Data is queried from the source document by using XPath expressions.

XML Firewalling

The Enterprise Gateway can detect intrusion in the following categories:

- XML content attacks
- XML Schema and DTD attacks
- Cryptographic attacks
- SOAP attacks
- Communication attacks; throttling



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

OEG is designed to capture incoming messages and to examine and evaluate them for possible attacks. This section describes some of the threats blocked by the XML firewall.

- XML content attacks:
 - Checking for XML well-formedness; XML document size; XPath and XQuery injection; SQL injection; XML encapsulation; XML viruses
 - Scanning outgoing messages for sensitive content based on metadata or regular expression patterns
 - Detecting XML bombs and XML clogging
 - Scanning WSDL files
- XML Schema and DTD attacks:
 - Checking for schema validation
 - Checking for XML entity expansion and recursion; schema poisoning; recursive elements, jumbo tag-names; malicious includes (also called “XML external entity [XXE]” attacks)

- Cryptographic attacks:
 - Checking for public key “Denial of Service”
 - Checking for replay attacks
- SOAP attacks:
 - SOAP operation filtering
 - Checking for rogue SOAP attachments (for example, viruses)
- Communication attacks:
 - HTTP header and query string analysis
 - IP address filtering
 - Traffic throttling

XML Content and Schema Attacks

Types of XML content and schema attacks:

- Denial of Service (DoS)
 - XML flooding
 - XML bombs
- Query injection
 - SQL injection
 - XPath/XSLT injection
- Message response filtering
- Schema “poisoning”
- XML viruses



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The slide lists the most common XML content attacks.

- The types of XML Denial of Service (DoS) include:
 - **XML flooding:** Exhausts the web services by sending a large number of legitimate SOAP messages
 - **XML bomb:** A block of well-formed and valid XML that crashes or hangs a program when that program attempts to parse it
- **Query injection:** Blocked through the detection of unexpected data within HTTP queries, including SQL injection and XPath injection
 - **SQL injection:** Malicious code is inserted into strings that are later passed to an instance of database for parsing and execution.
 - **XPath/XSLT injection:** Injection of expressions into the application logic
- **Message response filtering:** Large response messages are used for data harvesting attacks.
- **Schema “poisoning”:** The attack inserts an impostor schema reference into a SOAP message.
- **XML viruses:** Message attachments are used to transmit malicious executables, such as viruses or worms.

Denial of Service (DoS)

Attack	Countermeasure	Filter to Use
XML flooding	Impose data limits before parsing.	Throttling
XML bomb	Validate schemas to block attacks that make use of nested elements or excessive amounts of attributes.	XML Complexity
Large XML requests	<ul style="list-style-type: none">• Limit file size.• Block SOAP attachments.	<ul style="list-style-type: none">• Message Size• Content Type



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

DoS is an attempt to make the service endpoints unavailable to the intended users. The method of attack involves saturating the target service with external communication requests so that it cannot respond to legitimate traffic, or the service responds so slowly as to be rendered effectively unavailable.

The most common XML-related attacks are:

- **XML flooding:** Exhausts the web services by sending a large number of legitimate SOAP messages
- **XML bombs:** Overloading XML parser with nesting elements that are too deep, or putting too many elements in the XML instance
- **Large XML file size:** XML can wrap up any type of data, including multimedia or binary data. It is verbose by design in its mark-up of existing data and information. File size limits must be set high (up to hundreds of megabytes or gigabytes in size) or not limited at all. So it is possible for an attacker to execute a DoS attack by overloading the parser. Parsers based on the DOM model are especially susceptible to this attack because of the need to model the entire document in memory before parsing.

To protect your services against these types of attacks, take the countermeasures described in the table in the slide.

Example of XML Bomb: XML DoS DTD Recursion

DTDs are vulnerable to recursion attacks.

```
<?xml version="1.0" encoding="utf-8"?>
  <!DOCTYPE foobar [
    <!ENTITY x0 "hello">
    <!ENTITY x1 "&x0;&x0;">
    <!ENTITY x2 "&x1;&x1;">
    <!ENTITY x3 "&x2;&x2;">
    ...
    <!ENTITY x98 "&x97;&x97;">
    <!ENTITY x99 "&x98;&x98;">
    <!ENTITY x100 "&x99;&x99;">
  ]>
<foobar>&x100;</foobar>
```

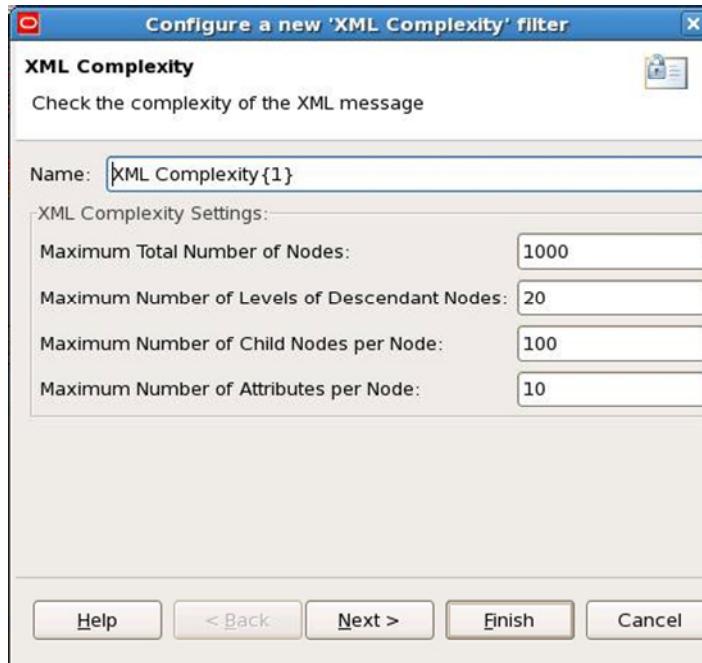


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The slide shows an XML bomb example that contains a recursively defined entity "&x100;".

Set Occurrences for Elements in XML Complexity Filter

- XML Schema supports the limiting of element occurrences.
- The Gateway goes further by enabling you to set limits on the “width” and “depth” of XML messages.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Parsing XML documents is a very processor-intensive activity. This can be exploited by hackers by sending large and complex XML messages to web services in a type of DoS attack that attempts to overload those services. The XML Complexity filter can protect against such attacks by performing the following checks on an incoming XML message:

- Checking the total number of nodes contained in the XML message
- Ensuring that the message does not contain deeply nested levels of XML nodes
- Making sure that elements in the XML message can contain only a specified maximum number of child elements
- Making sure that each element can have a maximum number of attributes

By performing these checks, the Enterprise Gateway can protect back-end web services from having to process large and potentially complex XML messages.

Note: You can also use the XML Complexity filter on the web service response to prevent a dictionary attack.

SQL Injection

- Categories of attack:
 - Web Application SQL Injection
 - Web Services SQL Injection
- Countermeasure: Verify SQL syntax.
- Filter to use: Threatening Content



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

SQL injection has the capability of influencing the SQL queries that an application passes to a back-end database. For example, an attack can insert SQL in XML to obtain more data than what the service was designed to return. With SQL injection, the attacker can leverage the syntax and capabilities of SQL itself, as well as the power and flexibility of supporting database functionality and operating system functionality available to the database.

There are two categories of SQL injection attacks:

- **Web Application SQL Injection:** Inserting SQL statements into web forms to force a database to return inappropriate data, or to produce an error that reveals database access information
- **Web Services SQL Injection:** For web services, this category of attack translates into manipulating data in a SOAP message to include SQL statements that will be interpreted by a back-end database.

To protect your applications and web services from these attacks, you can use SQL syntax verification.

Defending Against SQL Injection

SQL Injection Regular Expression

Name	Regular Expression
<input type="checkbox"/> Printf Format String Insertion	%(\d+\\$)?[-+0#]*(\\$ \d+)?(\.(\\$ \d+)?)*
<input checked="" type="checkbox"/> SQL Delete Attack	(?i)'[\s]*;[\s]*delete
<input checked="" type="checkbox"/> SQL Drop Table Attack	(?i)'[\s]*;[\s]*drop[\s]*table
<input checked="" type="checkbox"/> SQL Insert Attack	(?i)'[\s]*;[\s]*insert
<input checked="" type="checkbox"/> SQL Server Shutdown Attack	(?i)'[\s]*;[\s]*shutdown[\s]*with[\s]*nowa
<input checked="" type="checkbox"/> SQL Update Attack	(?i)'[\s]*;[\s]*update

Sample of SQL injection attack:

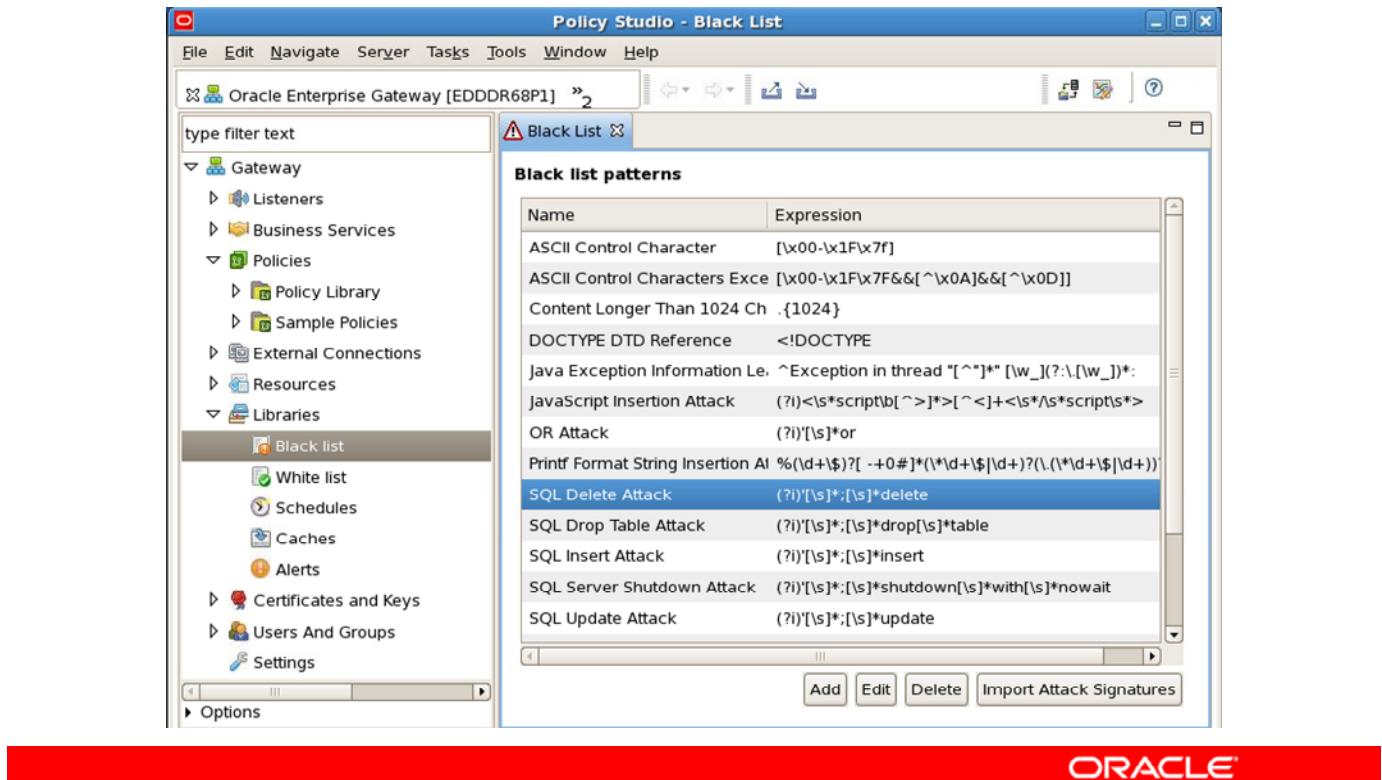
```
<ns:validateCard xmlns:ns="http://example.oracle.org/">
  <ns:cardType>
    ' ; delete usertbl;
  </ns:cardType>
  <ns:cardNr>1234-1234-1234-1234</ns:cardNr>
</ns:validateCard>
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The slide shows an example of SQL injection attack and the schema that counters the attack.

Threatening Content Filter



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Threatening Content filter can run a series of regular expressions that identify different attack signatures against request messages to check if they contain threatening content. Each expression identifies a particular attack signature, which can run against different parts of the request, including the request body, HTTP headers, and the request query string. In addition, you can configure the MIME types on which the Threatening Content filter operates.

The threatening content regular expressions are stored in the global Blacklist library. By default, this library contains:

- Regular expressions to identify SQL syntax to guard against SQL injection attacks (for example, SQL INSERT, SQL DELETE, and so on)
- DOCTYPE DTD references to avoid against DTD expansion attacks
- Java exception stack trace information to prevent call stack information getting returned to the client
- Expressions to identify other types of attack signature

XML Viruses

- Attack: SOAP attachments may contain viruses.
- Countermeasure: Scan attachments through an anti-virus engine.
 - Enterprise virus-scanning engine (McAfee, Sophos)
 - Open source virus-scanning engine (ClamAV)
- Note: These engines are supported by OEG (but, in the case of Sophos and McAfee, are separately licensed).
- Filters to use:
 - McAfee Anti-Virus
 - Sophos Anti-Virus
 - Clam Anti-Virus



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

XML viruses use SOAP with attachments or other attachment mechanisms to transmit malicious executables, such as viruses or worms. The OEG provides integration with several anti-viruses, either directly (in memory) or remotely via the Internet Content Adaptation Protocol (ICAP). McAfee, Sophos and ClamAV are supported out of the box.

The Enterprise Gateway supports three anti-virus engines:

- **McAfee Anti-Virus:** Scans incoming HTTP requests and their attachments for viruses and exploits. For example, if a virus is detected in a MIME attachment or in the XML message body, the Enterprise Gateway can reject the entire message and return a SOAP Fault to the client. In addition, this filter supports the cleaning of messages from infections such as viruses and exploits. It also provides scan type presets for different detection levels, and reports overall message status after scanning.
- **Sophos Anti-Virus:** Uses the Sophos Anti-Virus Interface (SAVI) to screen messages for viruses. You can configure the behavior of the Sophos library by using configuration options available in the Sophos Anti-Virus filter.
- **Clam Anti-Virus:** The Enterprise Gateway can check message attachment for viruses by connecting to a ClamAV daemon running on the network. The ClamAV daemon inspects the message. If the daemon finds a virus, it returns a corresponding response to the Enterprise Gateway, which can then block the message if necessary.

Message Response Filtering

- Scenarios:
 - Unusually large response messages from web services can indicate a data-harvesting attack.
 - In the event of a system crash, application and platform information could be sent back, which could be used in future attacks.
- Filters to use:
 - Message Sizing
 - Schema Validation



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Enterprise Gateway can check that XML messages conform to the structure or format expected by the web service by validating those requests against XML Schemas. An XML Schema precisely defines the elements and attributes that constitute an instance of an XML document. It also specifies the data types of these elements to ensure that only appropriate data is allowed through to the web service.

For example, an XML Schema might stipulate that all requests to a particular web service must contain a `<name>` element that contains, at most, a ten-character string. If the Enterprise Gateway receives a message with an improperly formed `<name>` element, it rejects the message.

You can find the Schema Validation filter in the Content Filtering category of filters in the Policy Studio. Drag the filter to the policy where you want to perform schema validation.

In addition, it is sometimes useful to filter incoming messages based on not only the content of the message, but also on external characteristics of the message. To do this, the Enterprise Gateway can be configured to reject messages that are greater than or less than a specified size.

Schema “Poisoning”

- Attack: An impostor schema reference is inserted in a SOAP message.
- Schema manipulation or alteration: Blocked through the usage of trusted schemas and not schemas in messages
- Countermeasure: Ignore schemas that are referenced from incoming SOAP messages, and use pre-existing schemas instead.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Schema “poisoning” occurs when an attacker inserts an impostor schema reference in a SOAP message

Cryptographic Attacks

- Replay attacks
- Dictionary attacks



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Replay attacks and dictionary attacks are common cryptographic attacks. They try to gain unauthorized access to the resources.

- **Replay attacks:** The attacks simply replay a valid message attempting to make unauthorized access to a system.
- **Dictionary attacks:** The attack breaks into a password-protected system by systematically entering every word in a dictionary as a password. It is blocked through detection of repeated failed authentication attempts.

Replay Attacks

- Attack: A valid message is replayed in an attempt to gain unauthorized access.
- Countermeasure: Use time stamps to block replay attacks. WS-Security includes support for time stamps.
- Filter to use: Validate Timestamp



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Suppose that a web service is being protected by an XML Gateway that scans incoming requests for X.509 certificates contained within SOAP messages, and to make sure the messages are encrypted and signed. This system is vulnerable to a replay attack that simply replays a valid message, thereby gaining unauthorized access.

The action you can take to protect your system against the attack is to use time stamps to block replay attacks. WS-Security includes support for time stamps. A replayed message will include the same time stamp as the original message. This means that both messages must be discarded, because it cannot be established which message is the original and which is the copy.

Caution: Beware of any solution that claims, “This is secure because all incoming messages are signed.”

Note: Do not confuse replay attacks with “flooding” DoS attacks.

REST: Overview

- What is REST?
 - Architectural style for state-driven applications
 - Constrained interaction model
- Why is REST used?
 - Move toward resource-based application model
 - Interoperability among Cloud, on-premise, and cross-domains
- Where is REST used?
 - RIA (AJAX, mashups), application services (alternative to SOAP), data services
 - Cloud services (APIs)



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

There are currently two architectural styles for developing web services: SOAP and REST.

REST is an architecture that uses the strengths of the web to build services. It proposes a set of constraints that simplify development and encourage scalable design. The REST style of designing web services relies on named resources rather than messages. The resources are uniquely identified by URIs. For example, in a RESTful book store system, each book order has a unique URI.

RESTful services mostly use HTTP as the protocol. Resource manipulation is based on the correct use of the HTTP verbs GET, POST, PUT, and DELETE.

REST components manipulate resources by exchanging representations of those resources. For example, a book order resource can be represented by an XML document. The book order might be updated by posting an XML document containing the changed purchase order to its URI.

Most “REST style” web services make heavy use of HTTP GET and do not use the other HTTP verbs. This means that, in practice, REST means “web services that are invoked by HTTP GETs with parameters in query strings.” REST is the building block of Web 2.0.

REST can be an alternative to SOAP, with a much simpler interface. However, it has some limitations that are related to the protocol: with REST, you are bound to HTTP. So it is difficult to handle different transports like MQ or JMS. As a result, if you are designing an application to be used exclusively on the web, REST might be the better option. In addition, REST design is good for database-driven applications and for clients who want quick integration.

Protecting REST

- Problem: REST stacks have difficulty accommodating sophisticated security requirements.
- Countermeasures:
 - Check the HTTP method used in the request.
 - Configure filtering rules for the URL query string.
 - Validate each of the query string parameters against a set of restrictive conditions.
- Filters to use:
 - Validate REST
 - Validate Query



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

RESTful applications simply rely on the built-in HTTP security (the HTTPS protocol). Unlike SOAP, you can have more sophisticated choices, such as WS-Security that lets you encrypt only portions of the messages and moves security handling from the transport layer to the message layer.

In Web 2.0, data is sent from the browser to the server, unknown to the user, in URL query strings: `getData.asmx?parameter=value`.

This means that many standard web application security techniques are applicable to REST web services:

- Validating the size of parameters on the query string
- Validating the content of parameters on the query string
- Examining parameters in the query string for known attacks such as SQL Injection
- Applying regular expressions (RegEx's) to query string parameters

Summary

In this lesson, you should have learned how to:

- Describe the different types of XML threats
- Identify the filters that block specific threats



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 8 Overview: Blocking XML Attacks

This practice covers the following topics:

- Applying XML Blocking Policy to service
- Testing with Service Explorer
- Protecting REST



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ClamAV has been installed and configured on the your lab machine and you will use it to scan viruses in message attachments.

Accelerating XML and Managing Traffic

9

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe how caching improves performance
- Configure caches in OEG
- Manage traffic



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Roadmap

- Using cache to accelerate XML processing
- Managing traffic with:
 - Throttling filter
 - Time filter



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Caching: Overview

- What: The request and its corresponding response
- Where: Memory or disk
- Why: Caching improves performance by:
 - Saving the trip of the message to the web service
 - Sparing the processing power on generating the same list
 - Avoiding querying the database and collecting the results repeatedly



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In cases where a web service is serving the same request (and generating the same response) over and over again, it makes sense to use a caching mechanism. When a cache is employed, a unique identifier for the request is cached with the corresponding response for this request. If an identical request is received, the response can be retrieved from the cache instead of forcing the web service to reprocess the identical request and generate the same response. The use of caching in this way helps divert unnecessary traffic from the web service and makes it more responsive to new requests.

This approach results in the following performance improvements:

- The Enterprise Gateway does not have to route the message to the web service, therefore saving the processing effort required and, perhaps more importantly, saving the time it takes for the round trip.
- The web service does not have to waste processing power on generating the same list over and over again, therefore making the service more responsive to requests for other services.

- Assuming a naive implementation of database retrieval and caching, the web service does not have to query the database (over the network) and collate the results repeatedly for every request.

The caching mechanism used in the Enterprise Gateway offers full control over the size of the cache, the lifetime of objects in the cache, whether objects are cached to disk or not, and even whether caches can be replicated across multiple instances of Enterprise Gateways.

Using Caching

The Enterprise Gateway uses caching for a variety of purposes:

- Caching response messages
- Caching attributes looked up from databases, directories, and other sources
- Caching login session data (for example, SSO tokens)
- Keeping a count for throttling filters

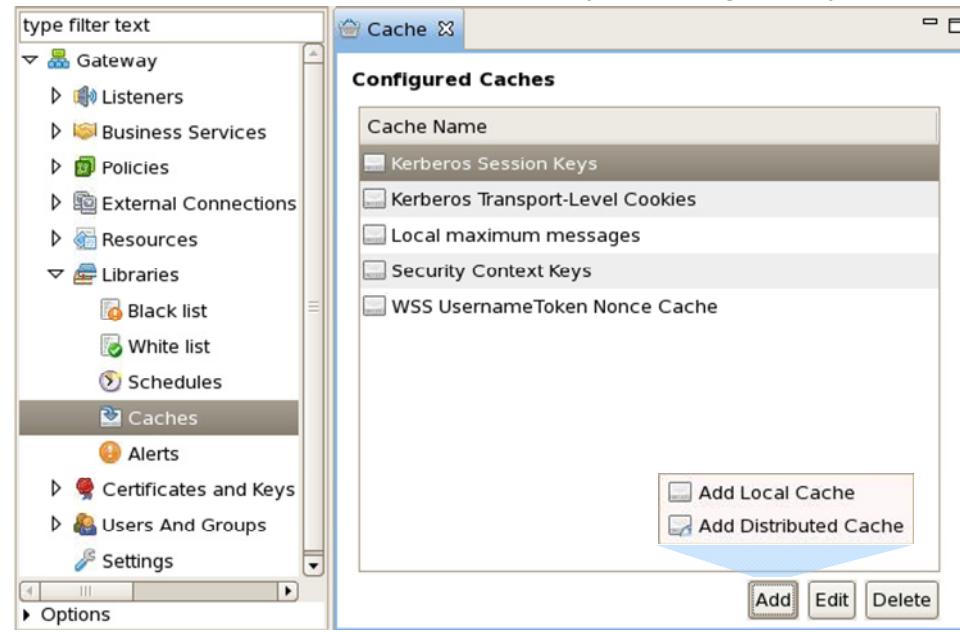


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Configuring Caches

You can configure:

- Local cache: A single Enterprise Gateway instance
- Distributed caches: Multiple Gateways throughout your network



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

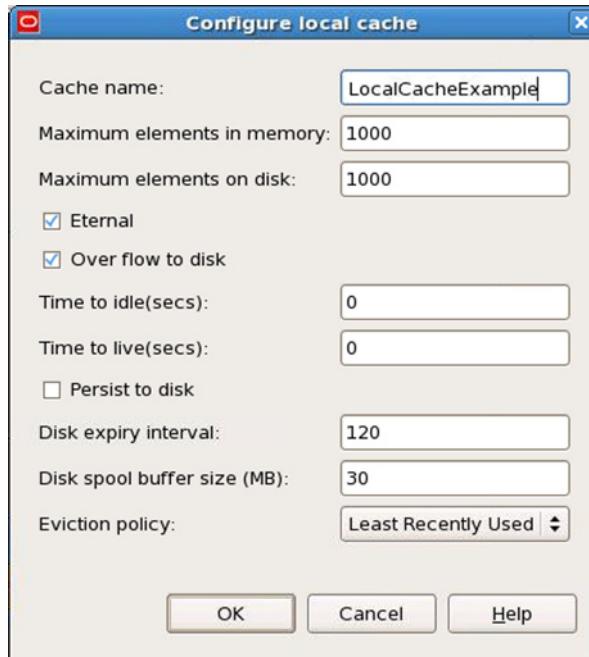
You need to configure a cache before creating a policy to use it. You can configure both local and distributed caches in the Enterprise Gateway.

- **Local cache:** The local cache is used where a single Enterprise Gateway instance has been deployed. In such cases, you do not need to replicate caches across multiple running instances of the Enterprise Gateway.
- **Distributed cache:** If you have deployed several Enterprise Gateways throughout your network, you need to employ a distributed cache. In this scenario, each Enterprise Gateway has its own local copy of the cache but registers a cache event listener that replicates messages to the other caches so that put, remove, expiry, and delete events on a single cache are duplicated across all other caches.

In Policy Studio, you add a local cache by selecting the top-level Caches tree item on the External Connections tab and then clicking the Add button at the bottom-right of the screen. Select “Add Local Cache” from the menu.

Similarly, you configure a distributed cache by selecting the top-level Caches tree item on the External Connections tab and then clicking the Add button at the bottom-right of the screen. Select “Add Distributed Cache” from the menu.

Configuring Local Cache Settings



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can configure the following fields in the Configure Local Cache dialog box:

- **Cache name:** Enter a name for the cache.
- **Maximum elements in memory:** Enter the maximum number of objects that can be in memory at any one time.
- **Maximum elements on disk:** Set the maximum number of objects that can be stored in the disk store at any one time. A value of zero indicates an unlimited number of objects.
- **Eternal:** If this option is selected, objects stored in the caches never expire and timeouts have no effect.
- **Over flow to disk:** Select this option if you want the cache to overflow to disk when the number of objects in memory has reached the amount set in the “Maximum elements in memory” field.

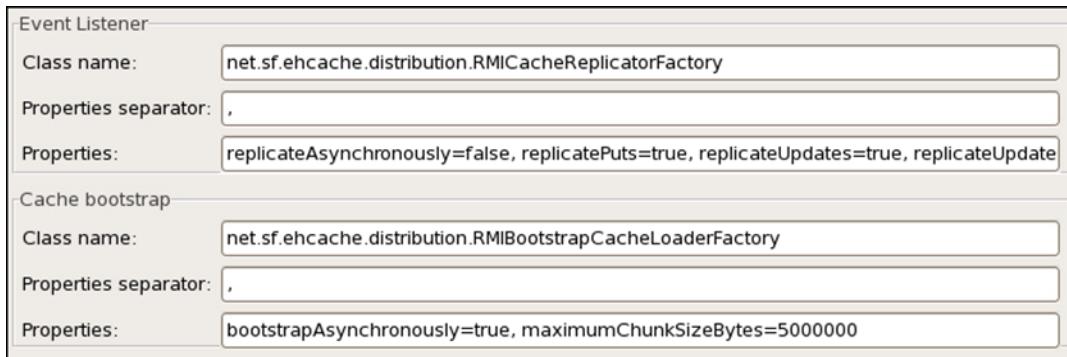
The remaining fields are optional. See more details in the online Help.

Cache configuration can persist across restarts of a gateway.

Configuring Distributed Caches

Additional settings for distributed caching include replication settings:

- TCP Multicast, which is used to discover peers by default
- Can manually list other Gateways if necessary



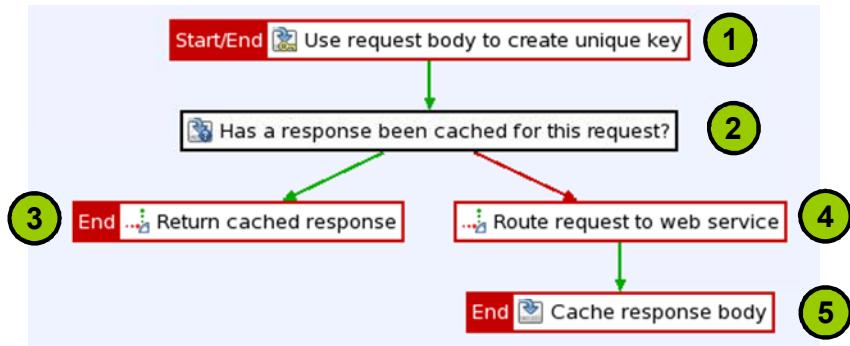
ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Many of the settings for the distributed cache are identical to those for the local cache. The screenshot shows the additional settings that are specific to distributed caching.

- **Event Listener:** The cache needs to register listeners for cache events, such as put, remove, delete, and expire. The fields in this section enable you to configure the listener.
- **Cache bootstrap:** The cache can call on a factory class when it starts to prepopulate itself. The fields in this section enable you to configure the factory class.

Caching Response Messages: Example



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

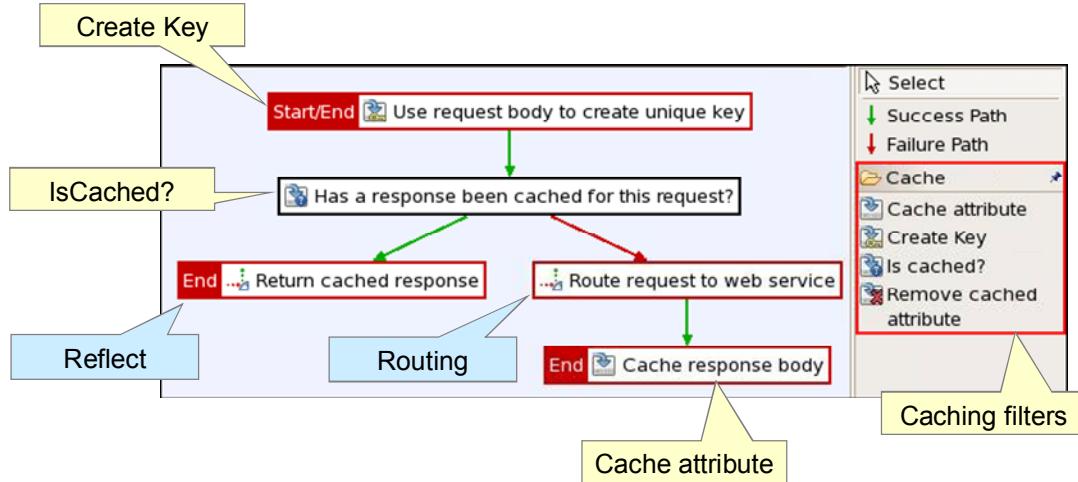
After the cache is configured, you can create caching policies that use the cache. The example in the slide shows the circuit that caches responses from the web service. It uses the request body to identify identical successive requests. When the Enterprise Gateway receives two successive requests with identical message bodies, it returns the corresponding response from the cache instead of routing the request to the web service.

The logic of the circuit is summarized as follows:

1. The first filter configures the part of the request that you want to use to identify unique requests. This example uses the request body as the unique key, which is then used to look up the appropriate response message from the cache.
2. The second filter looks up the request body in the response cache to see if it contains the request body.
3. If the response cache contains the request body, the response message that corresponds to this request is returned to the client.
4. If the response cache does not contain the request body, the request is routed to the web service, which processes it (by connecting to a database over the network and running a SQL statement) and returns a response to the Enterprise Gateway.

5. The Enterprise Gateway returns the response to the client and caches it in the response cache. When the next identical request is received by the Enterprise Gateway, the corresponding response is located in the response cache and returned immediately to the client.

Caching Filters



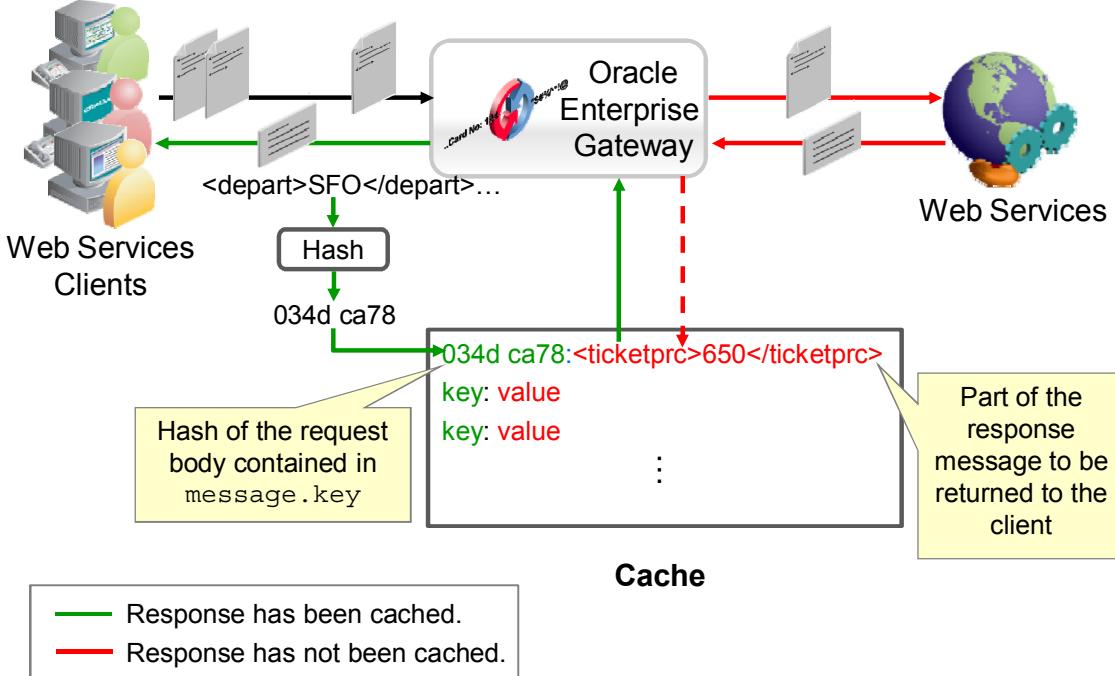
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To implement the sample circuit, you must use the following filters:

- **Create Key:** To identify the part of the message that determines whether a message is unique
- **Is Cached?:** To look up a named cache to see if a specified message attribute has already been cached
- **Reflect:** To return the cached response to the client if the IsCached? filter passes
- **Routing:** If a response for this request could not be located in the cache, the Enterprise Gateway routes the request to the web service and waits for a response.
- **Cache Attribute:** To configure the part of the message that you want to cache

How Caching Works with Filters



ORACLE

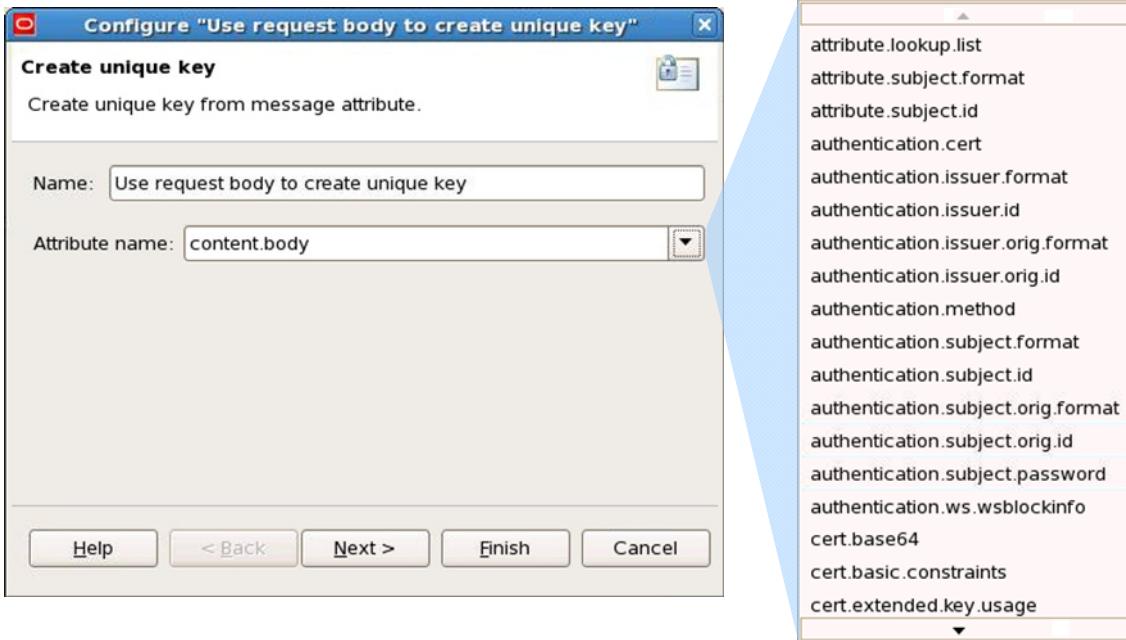
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The diagram depicts how caching works with the filters. After the part of the request is specified for unique identification, the OEG hashes the message and uses it as the key for the object in the cache. The cache contains a list of key-value pairs. The key is the hashed message part, which is contained in the `message.key` attribute, and the value is the cached response for the request to be returned to the clients.

The filter looks up the cache by using the key to see if a response has been stored for the current request.

- If the key is found in the cache, the value of the key is returned to the client.
- If the key is not found, the request is routed to the web service, which processes it and returns a response to the Gateway. The Gateway caches the response for future use.

Create Key Filter



ORACLE

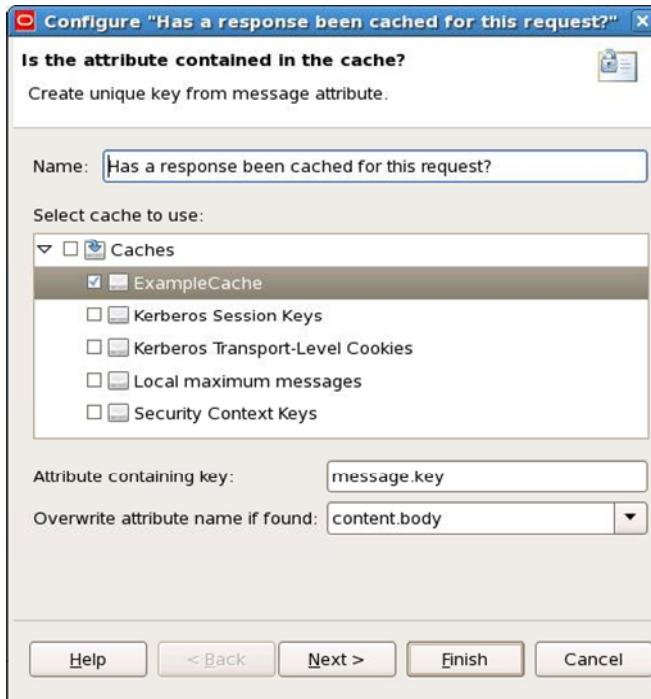
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This filter is used to decide what part of the request is used for a request to be considered unique. Different parts of the request can be identified internally using Oracle message attributes. For example, `content .body` contains the request body.

In the example in the slide, the fields are configured as follows:

- **Name:** Use request body to create unique key
- **Attribute name:** content .body

“IsCached?” Filter



ORACLE®

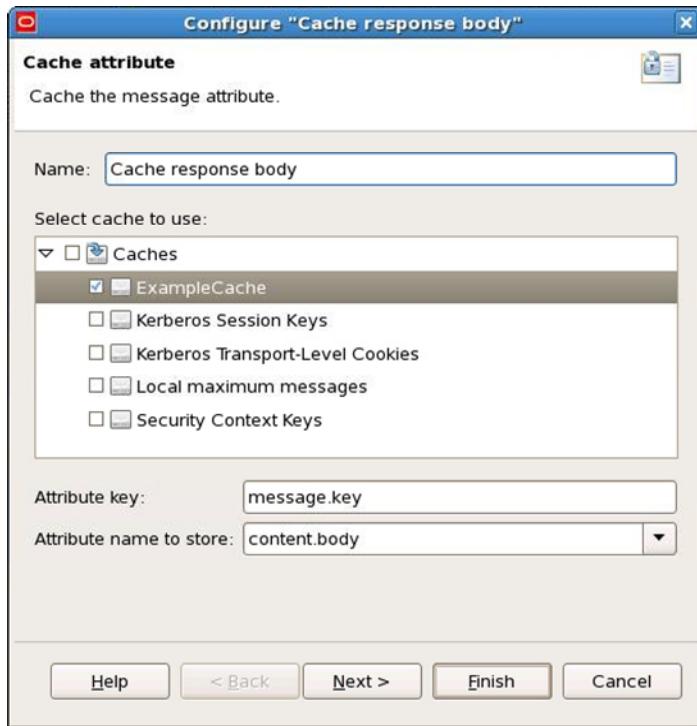
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This filter looks up the cache to see if a response has been stored for the current request. It looks up the cache by using a message attribute, which is `message.key` by default. The `message.key` attribute contains the key for objects in the cache. If the key is found in the cache, the value of the key (cached response for this request) is written to the `content.body` attribute, which can be returned to the client by using the Reflect filter.

In the example in the slide, the fields are configured as follows:

- **Name:** Has a response been cached for this request?
- **Select cache to use:** ExampleCache (assuming you have created a cache of this name)
- **Attribute containing key:** `message.key`
- **Overwrite attribute name if found:** `content.body`

“Cache Attribute” Filter



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When the response has been received from the web service, it should be cached for future use. The Cache Attribute filter is used to configure the key that is used to look up the cache and to configure which aspect of the response message is stored as the key value in the cache.

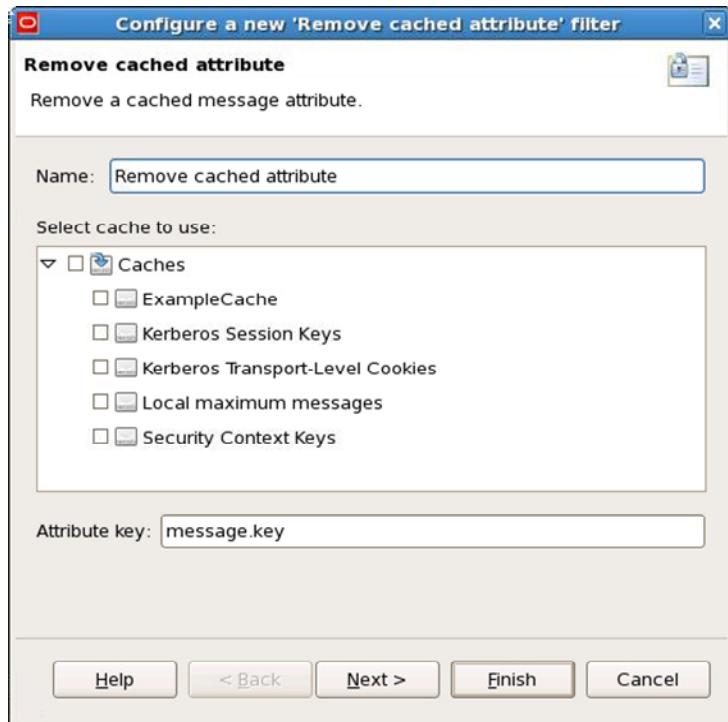
Note that the example in the slide specifies the value of the `content.body` attribute to cache. Because this filter is configured after the routing filters, this attribute contains the response message.

Note also that the value entered in the “Attribute key” field should match that entered in the “Attribute containing key” field in the IsCached? filter.

In the example in the slide, the fields are configured as follows:

- **Name:** Cache response body
- **Select cache to use:** ExampleCache
- **Attribute key:** message.key
- **Attribute name to store:** content.body

“Removed Cached Attribute” Filter



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Remove Cached Attribute filter enables you to delete a message attribute value that has been stored in a cache. Each cache is essentially a map of name-value pairs, in which each value is keyed to a particular message attribute. For example, it is possible to store a cache of request messages according to their message IDs. In this case, a message's ID attribute is the key into the cache, which stores the value of the request message's `content.body` message attribute.

The preceding example may be useful in cases where a request message may need to be cached and stored until the request has been fully processed and a response returned to the client. For example, if the request must be routed to a back-end web service, but that web service is temporarily unavailable, it may be possible to configure the circuit to resend the cached request instead of forcing the client to retry.

Quiz

Which of the following filters is used to configure the part of the message that you want to cache:

- a. Is Cached?
- b. Create Key
- c. Cache Attribute
- d. Set Attribute



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

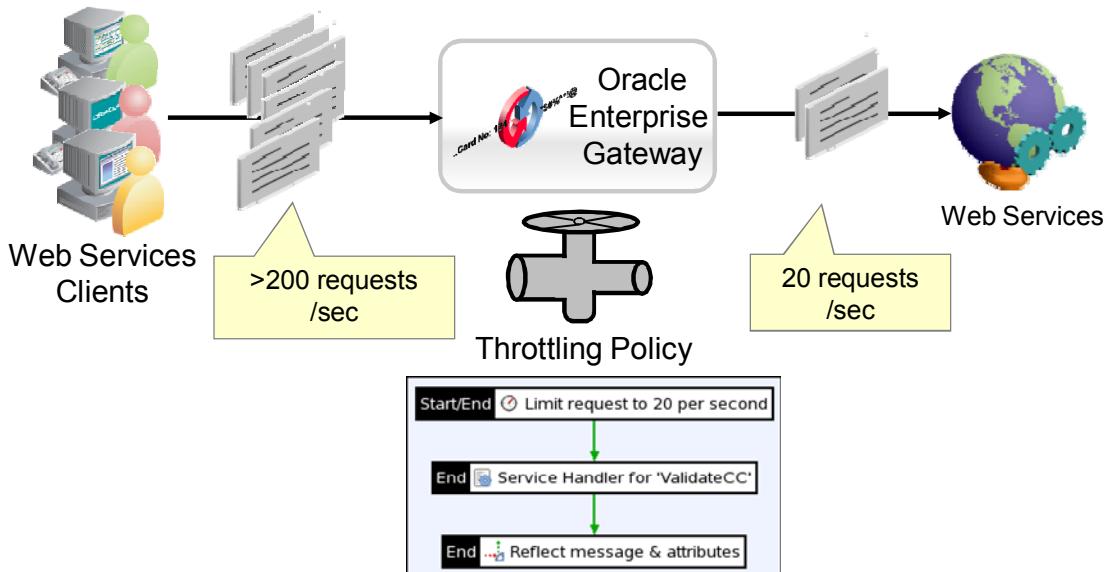
Roadmap

- Using cache to accelerate XML processing
- Managing traffic with:
 - Throttling filter
 - Time filter



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Traffic Throttling



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

OEG protects web services from unanticipated traffic spikes by smoothing out the traffic. This typical OEG use case is called throttling. You can configure the filter to allow only a certain number of messages from a specified client to go through to the protected system in a given time frame.

If the number of messages exceeds the specified limit, the filter fails for the excess messages. It is important to note that the filter still succeeds for incoming messages that meet the specified constraints. For example, if the filter is configured to allow 20 messages through per second, it fails for the 21st message but passes for the first 20 incoming messages. When the filter detects a higher number of incoming requests, it blocks the messages.

Throttling restricts incoming connections and the number of messages to be processed.

Throttling can be applied to PoX, SOAP, REST, or any payload or request. It is also independent of the protocol used. Traffic can be regulated at a single gateway or cluster-of-gateways level. You can apply traffic restriction rules for a service, an operation, or even time of day. Those restrictions can be applied depending on the service name, the user identity or IP address, content from the payload, protocol headers, or any combination of the preceding elements.

Benefits of Throttling

- Protects back-end systems from excessive load
- Prevents “Denial of Service” (DoS) attacks
- Enforces and monitors service consumption
- Assists with capacity planning



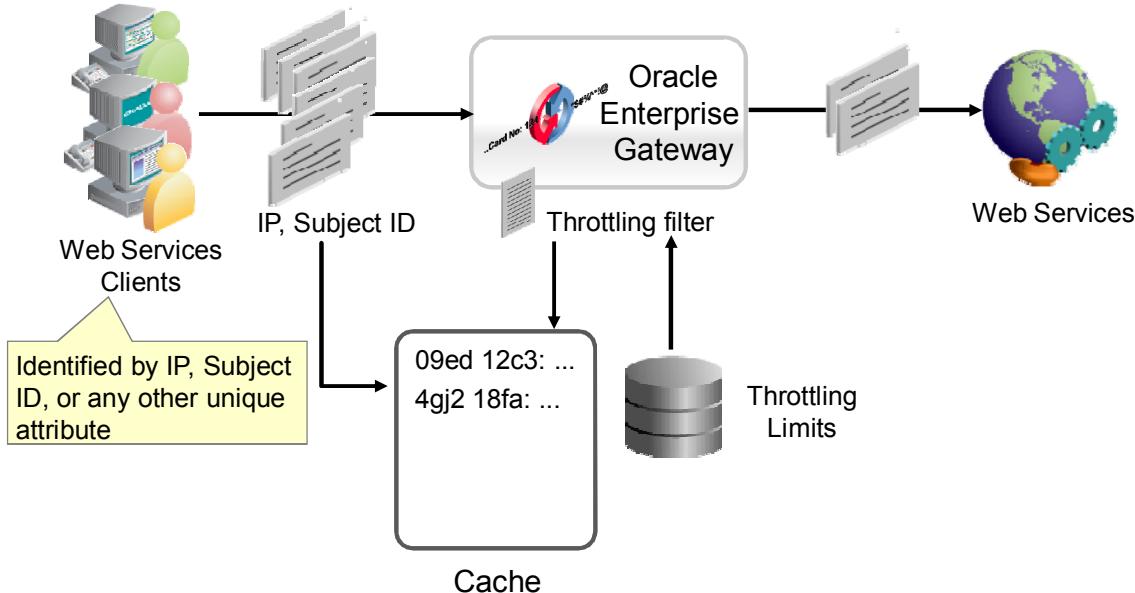
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Throttling protects back-end systems from message flooding and prevents DoS attacks proactively and pre-emptively. It also limits clients to agreed web service consumption levels in accordance with service usage agreements. This enables Oracle customers to charge their clients for different levels of web service usage. With throttling functionality, you can identify traffic surge patterns to help you perform capacity planning.

However, throttling is NOT:

- Buffering or “Traffic Smoothing” – This involves temporary persistence of requests exceeding a spike limit and applies more to a guaranteed delivery design.
- Guaranteed to prevent downstream service overload – Sizing and capacity limits must be established or estimated for target systems.

How Throttling Works



ORACLE®

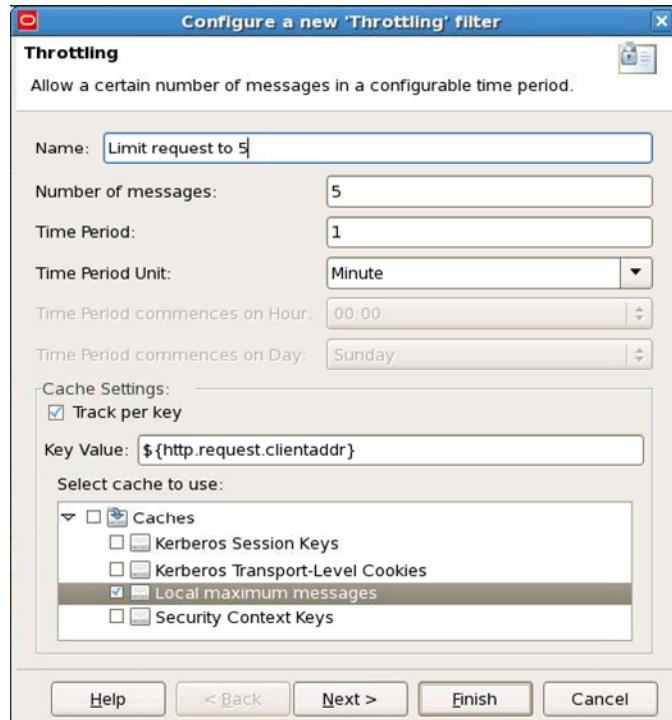
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can configure the Enterprise Gateway to keep track of request messages based on a specific key value. In other words, if more messages that match this key are received than are allowed, the filter fails.

Throttling limits can be kept in an external source, such as a database or service repository. The gateway retrieves limit values and caches them for efficiency. Throttling filters are configured to use those values. The cache is flushed regularly, and data is read from the database repository again.

Configuring a Throttling Filter

- Requires Number of Messages, Time Period, and Time Period Unit (values can be message attributes)
- Use “Track per key” setting to track by IP, PartnerID, SubjectID, and so on.
- Use local or distributed cache to save the data.



ORACLE

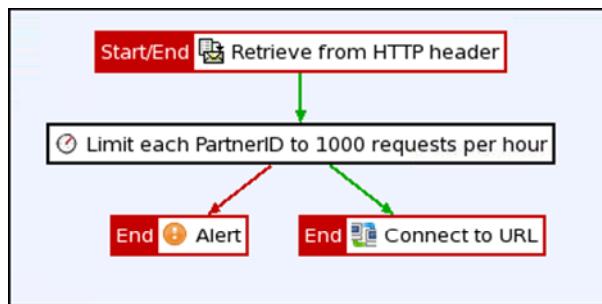
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When creating a Throttling filter, you must configure the general settings and cache settings.

- **Track per key:** Select this check box if you want to configure the gateway to keep track of request messages based on a specific key value.
- **Key Value:** You can perform message filtering based on a particular key. This key is used to look up entries in the cache you defined. The key entered can be a combination of a fixed string value and/or Enterprise Gateway message attribute. For example, the following key can be used to keep track of the number of times a particular URI is requested:
`MSG_COUNT-${http.request.uri}`
- **Select cache to use:** The Throttling filter uses the preconfigured “Local maximum messages” cache by default. You can configure more caches by using the Global Cache interface.

Handling Throttling Violations

- Raise an alert (email, SNMP, syslog, and so on).
- Route overflow requests to a secondary server.
- Return a particular message to the client with a configurable HTTP code.
- Place the message on a message queue to be read at a later time by OEG's message queue reader.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

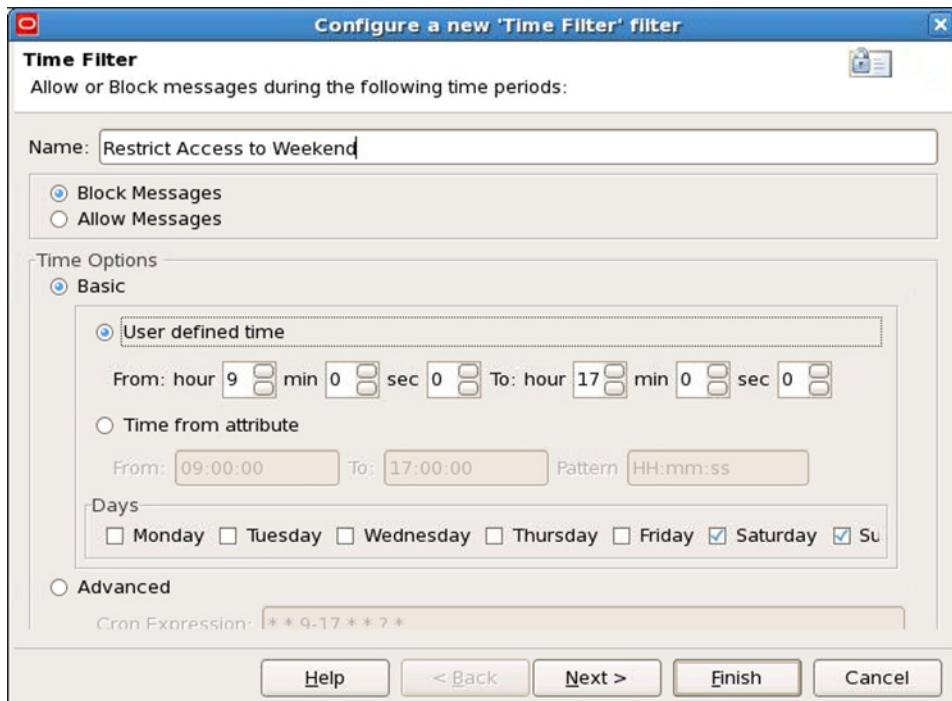
The Enterprise Gateway's behavior in the case of a breach in the configured constraints is determined by the filter that is next in the failure path for the Throttling filter in the policy.

Possible actions when throttling reaches the limit include the following:

- Raise an alert (email, SNMP, syslog, and so on).
- Route overflow requests to a secondary server (for scenarios where a particular server can process only a certain number of concurrent requests).
- Return a particular message to the client (for example, Server limit has been reached. Please try again.) with a configurable HTTP code.
- Place the message on a message queue to be read at a later time by OEG's message queue reader. In this way, the message can be processed at a later time when the concurrent message count is smaller.

In the example in the slide, an Alert filter is configured as the successor filter in the failure path in the policy.

Time Filter



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Time filter is another filter that can be used to manage traffic. The Time filter enables you to block or allow messages at a specified time of day and/or on a specified day of the week. You can input the time of day directly in the Time Filter screen, configure message attributes to supply this information by using the Java SimpleDateFormat, or specify a cron expression for advanced setup.

You can use the Time filter in any policy circuit (for example, to block messages at specified times and/or on days when a web service is not available, or when the service has not been subscribed for by a consumer). In this way, the filter enables you to meter the availability of a web service and enforce service-level agreements.

Quiz

Identify the component that must exist when configuring the throttling filter:

- a. Database
- b. Service repository
- c. Cache
- d. Message queue



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Summary

In this lesson, you should have learned how to:

- Describe how caching improves performance
- Configure caches in OEG
- Manage traffic with filters



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 9 Overview

This practice covers the following topics:

- Caching response messages from the service
- Applying Gateway-wide (global) throttling
- Applying throttling at service level



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

10

Configuring SSL

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe how OEG manages certificates and keys
- Describe SSL support in the Enterprise Gateway
- Set up SSL
- Set up mutual SSL



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

SSL Basics

- SSL provides a secure channel where the server and the client are explicitly authenticated.
- SSL uses signed digital certificates from a certificate authority (CA) for authentication.
- There are two modes of SSL authentication:
 - Server authentication: One-way SSL authentication that requires only a server certificate
 - Client authentication: Two-way SSL authentication that requires client certificates for a mutually authenticated channel



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Secure Sockets Layer (SSL), also called Transport Layer Security (TLS), is the most commonly used protocol to secure communications between a web browser and a server. SSL uses signed digital certificates from a certificate authority (CA) for authentication. Signed digital certificates enable two entities connecting in a network to authenticate each other's identity. To ensure maximum security, a certificate is issued by a third-party CA.

There are two modes of SSL authentication:

- **Server authentication:** A one-way SSL authentication enabling the SSL client to verify the identity of the application operating as the SSL server. For example, when you access your brokerage website and log in through a web form, you are using server authentication.
- **Client authentication:** A two-way SSL authentication that the client needs to verify the identity of the server. It must also provide a digital certificate to authenticate itself with the server. The protocol involves challenge and response between the server and the client in which information is digitally signed to authenticate possession of the private key.

Digital Certificate

A digital certificate is:

- A data structure that contains identity information
 - Organizational information
 - Public key
 - Digital signature
 - Expiration date
- Issued by a third-party known as a certificate authority



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A digital certificate is a standard means of verifying that an entity such as a server, client, or application is who (or what) it claims to be. To ensure maximum security, a certificate is issued by a third-party CA such as VeriSign or Thawte.

A digital certificate contains the following information to verify the identity of an entity:

- **Organizational information:** It is information that uniquely identifies the owner of the certificate, such as organizational name and address.
- **Public key:** The receiver of the certificate uses the public key to decipher encrypted text sent by the certificate owner to verify its identity. A public key has a corresponding private key that encrypts the text.
- **Digital signature:** The issuer of the certificate signs it with a digital signature to verify its authenticity.
- **Expiration date:** A digital certificate is issued for a limited time. When its expiration date passes, the digital certificate must be replaced.

The web browser basically has a built-in list of all the main certification authorities and their public keys and uses that information to decrypt the digital signature. This enables the browser to quickly check for problems and abnormalities.

Chain of Trust

Digital certificates are verified by using a chain of trust.

Key Ring in Browser



A Server Digital Certificate



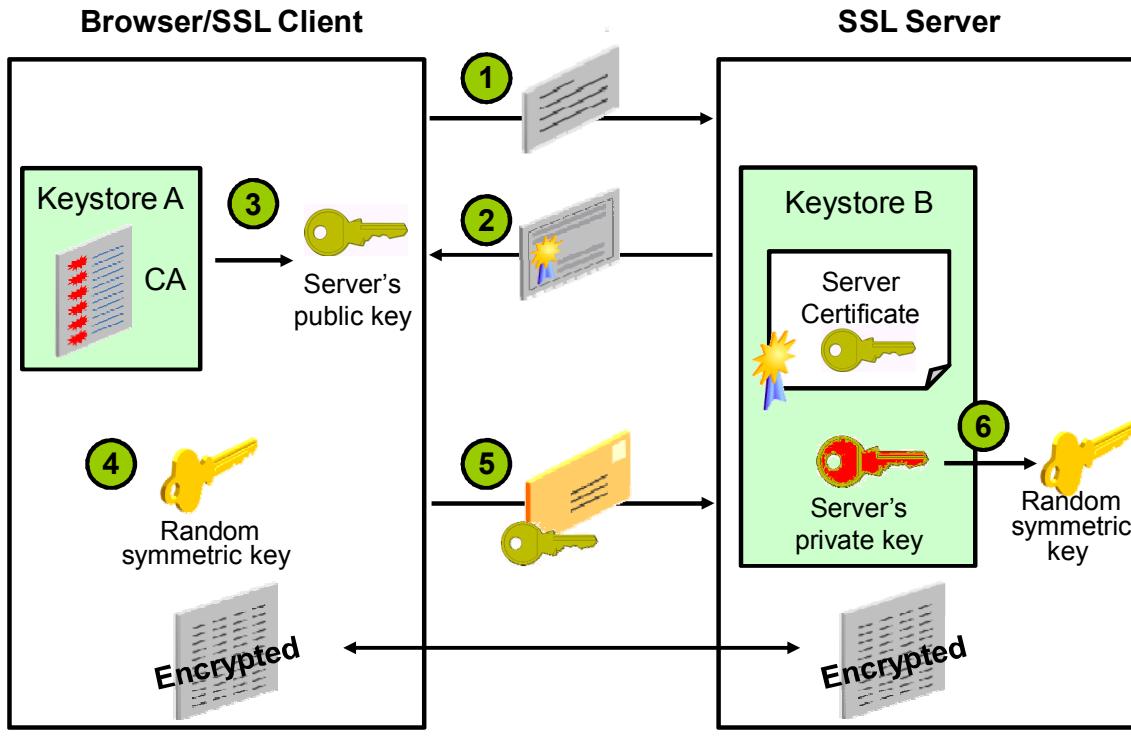
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A CA can issue multiple certificates in the form of a tree structure. A root certificate is the top-most certificate of the tree, which is used to sign other certificates. All certificates immediately below the root certificate inherit the trustworthiness of the root certificate.

End users must decide for themselves which CAs they will accept as being trusted. To be able to verify server digital certificates, client web browsers are required to possess the root CA certificates used by servers. Popular web browsers usually come with a key ring where several CA digital certificates, called *trusted roots*, are already installed. This list can be edited, and the digital certificates of untrusted CAs can be deleted.

SSL Handshake



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

An SSL connection is usually initiated by the client by using a URL starting with https://. An SSL handshake is performed at the beginning of an SSL session. SSL implements shared key encryption between its entities after first transporting the shared key via a public key.

The diagram in the slide depicts how the SSL handshake is processed:

1. The browser (SSL client) sends a secure session request message.
2. The server responds by sending the signed digital certificate to the client. If the server application requires a digital certificate for client authentication, the server sends a digital certificate request message.
3. The client uses the corresponding CA certificate stored in its keystore to authenticate the server's certificate.
4. After verifying the signature on the certificate, the client generates a random symmetric key.
5. The client encrypts the random symmetric key by using the server's public key, and sends it (contained in the message) to the server. If the server requested a client digital certificate, the client sends a digital certificate, or if no suitable digital certificate is available, the client sends a "no digital certificate" alert.

6. After receiving the client message, the server uses its private key to decrypt the message to retrieve the symmetric key.

When the SSL handshake ends, client and server both have the symmetric key and use it to encrypt application data for the duration of the session.

SSL Support in the Enterprise Gateway

- Multiple HTTPS interfaces can be created to listen for requests.
- Interfaces can be configured to require a client certificate (mutual SSL).
- The gateway can act as an SSL termination point.
- SSL is also available for other connections, such as LDAP, SMTP, Tibco EMS, and so on.
- The gateway supports the most common cipher suites. At connection time, it uses the strongest cipher suite supported by the client.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Enterprise Gateway supports the following features for SSL connections:

- You can create one or more HTTPS interfaces to listen for requests. When you add an HTTPS interface to an HTTP service group (for example, Default Services), all paths and services in the group will be available for both HTTP and HTTPS connections.
- The HTTPS interface can accept mutually authenticated SSL connections.
- As an SSL termination point, the gateway can convert encrypted data sent through an SSL session to unencrypted data.
- The most common use of SSL is to secure HTTP communication between a client and a server. However, OEG also supports SSL for other connections, such as LDAP, SMTP, Tibco EMS, and so on.
- During the SSL handshake, the client first sends the list of cryptographic suites that it supports with the request message before the server sends its digital certificate. After the client receives and verifies the server digital certificate, it realizes that the server digital certificate uses stronger encryption, so the client and server will then use these stronger cryptographic suites for transferring data.

Tips for Using SSL

- Add SSL only for specific paths and services.
- Use single SSL connection for multiple requests.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- SSL has a performance cost. It is recommended that you add SSL for only those paths and services that must be secured.
- When an SSL connection is established, an SSL handshake occurs. After a connection is made, SSL performs bulk encryption and decryption for each read/write. The performance cost of an SSL handshake is much higher than that of bulk encryption and decryption. As a result, using a single SSL connection to send multiple requests (rather than opening a connection for each request) can enhance SSL performance.

Quiz

Which of the following is NOT needed in the SSL handshake?

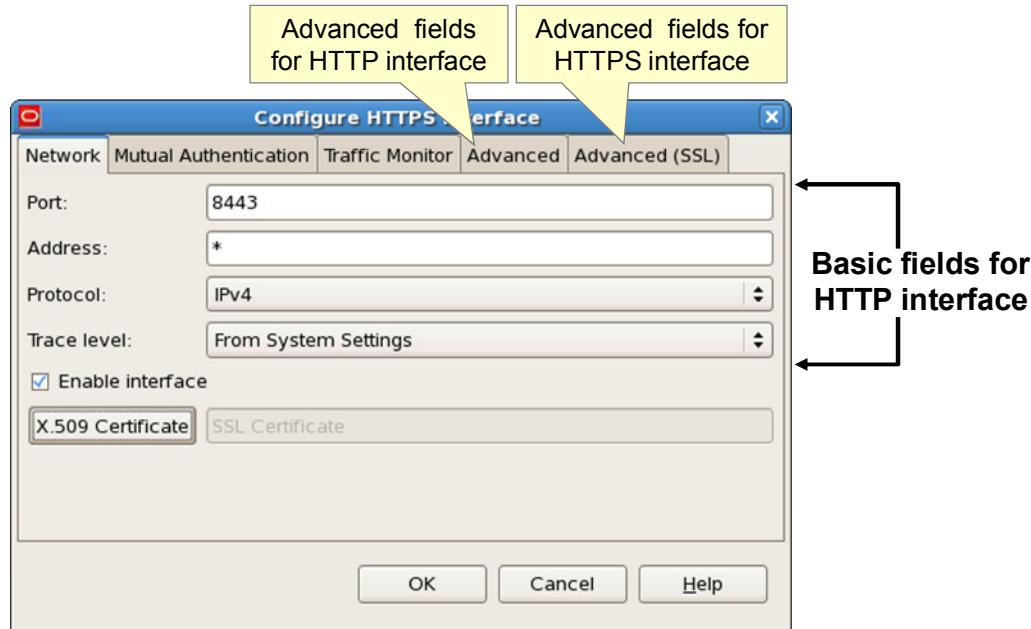
- a. Server digital certificate
- b. CA certificate
- c. Symmetric key
- d. Asymmetric key



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: d

Configuring the HTTPS Interface



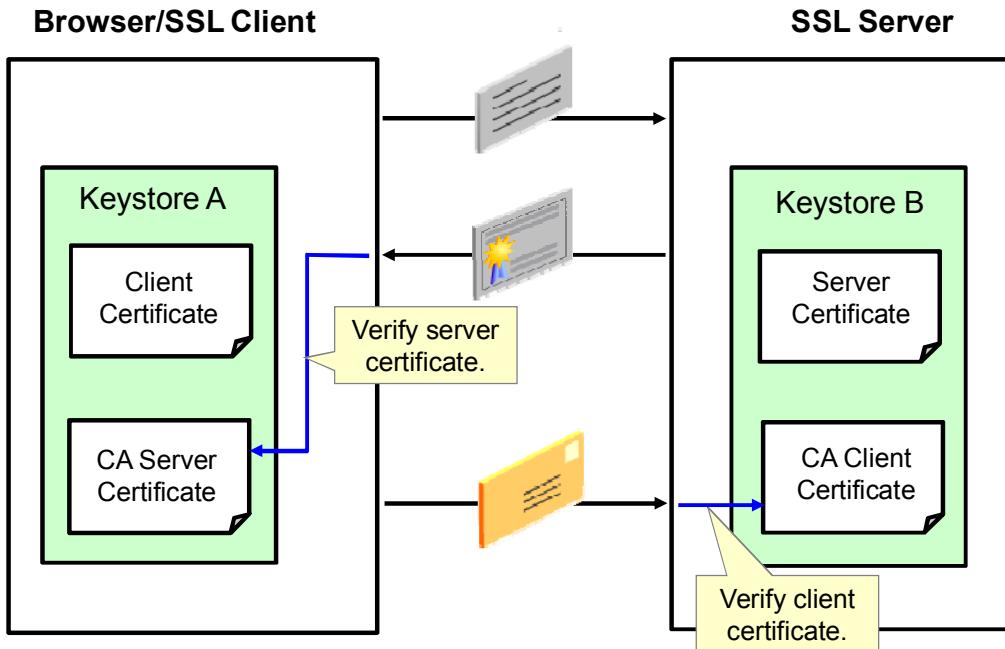
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To configure the HTTPS interface, you must complete the same fields for an HTTP interface on the Network tab, with the addition of the following setting:

X.509 Certificate: Select the certificate that the Enterprise Gateway uses to authenticate itself to clients during the SSL handshake.

Mutual SSL Authentication



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

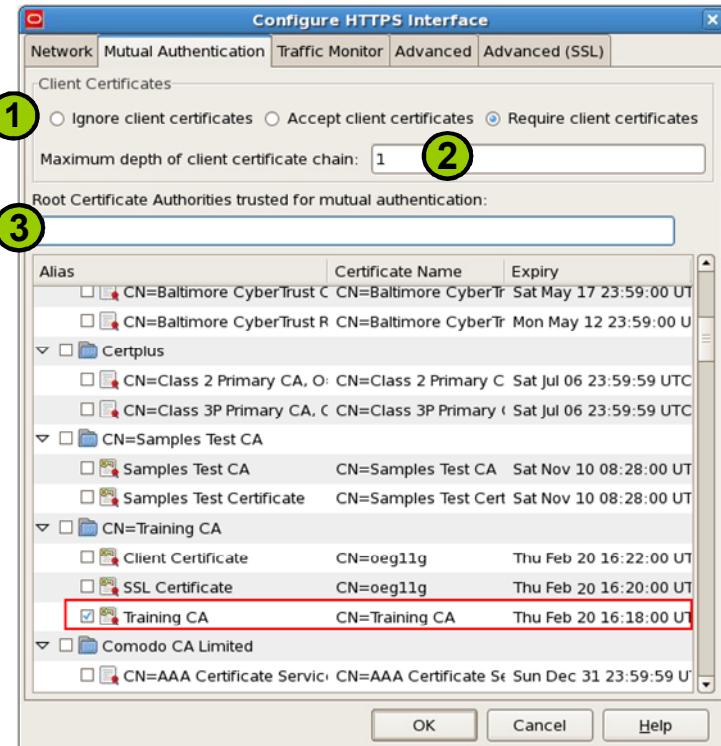
In mutual (two-way) SSL authentication, the SSL client application verifies the identity of the SSL server application, and then the SSL server application verifies the identity of the SSL-client application.

Client certificates are typically issued by a CA. In most cases, the CA includes a copy of its certificate in the client certificate so that consumers of the certificate can decide whether or not to trust the client, based on the issuer of the certificate.

It is also possible that a chain of CAs were involved in issuing the client certificate. For example, a top-level organization-wide CA (for example, Company CA) might issue department-wide CAs (for example, Sales CA, QA CA, and so on), and each department CA is then responsible for issuing a client certificate to all department members. In such cases, the client certificate may contain a chain of one or more CA certificates.

Configuring Mutual Authentication Settings

1. Choose to ignore, accept, or request client certificate.
2. Select one or more trusted authorities.
3. Select Root certificate authorities.



ORACLE

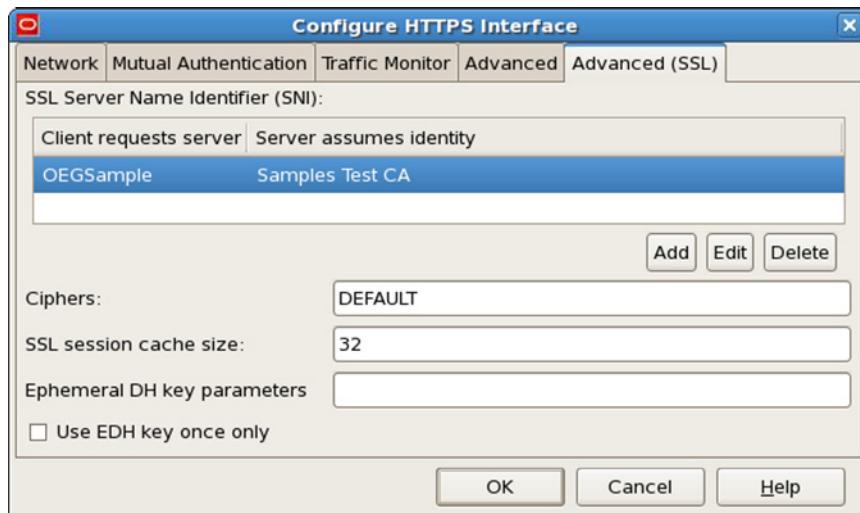
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Mutual SSL can be set on each HTTP interface. You can configure clients to authenticate to the Enterprise Gateway on the Mutual Authentication tab, where the following options are available:

- **Ignore client certificates:** The gateway ignores client certificates if they are presented during the SSL handshake.
- **Accept client certificates:** Client certificates are accepted when presented to the gateway, but clients that do not present certificates are not rejected.
- **Require client certificates:** The gateway accepts connections from only those clients that present a certificate during the SSL handshake.
- **Maximum depth of client certificate chain:** You can use this field to configure how many CA certificates in a chain of one or more are trusted when validating the client certificate. By default, only one issuing CA certificate is used, and this certificate must be checked in the list of trusted root certificates. If more than one certificate is used, only the top-level CA must be considered trusted, while the intermediate CA certificates are not.
- **Root Certificate Authorities trusted for mutual authentication:** Select the root CA certificates that the gateway considers trusted when authenticating clients during the SSL handshake. Only certificates that are signed by these selected CAs are accepted.

Configuring Inbound SSL Settings

- Set SSL session cache size to improve performance.
- Use SSL SNI to assign an identity to a server based on the incoming host name.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

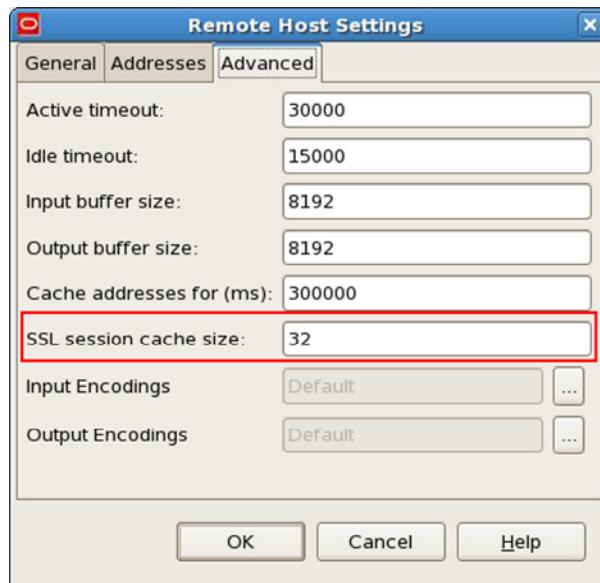
For incoming connections (client-to-gateway), you can set the SSL session cache size on the Advanced (SSL) tab, which is displayed in the slide. This controls the number of idle SSL sessions that can be kept in memory. You can use this setting to improve performance because it caches the slowest part of establishing the SSL connection. A new connection does not need to go through full authentication if it finds its target in the cache. The cache size defaults to 32. If there are more than 32 simultaneous SSL sessions, this does not prevent another SSL connection from being established, but it means that no more SSL sessions are cached. A cache size of 0 means the cache size is unlimited.

You can use SSL SNI to assign an identity to a server based on the incoming host name (useful for virtual hosting).

Note: This works only with modern browsers that are able to send the SNI in the request (prior to the SSL handshake).

Configuring Outbound SSL Settings

For outbound SSL connections, SSL session cache size is controlled at the remote host level.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can specify the size of the SSL session cache for outbound SSL connections to the remote host. This controls the number of idle SSL sessions that can be kept in memory. Like the inbound SSL session, this setting is used to improve performance. The default cache size is 32. A cache size of 0 means the cache size is unlimited.

Quiz

In mutual authentication, client certificates are typically issued by:

- a. A server certificate
- b. A CA certificate
- c. A root certificate
- d. A browser



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

In this lesson, you should have learned how to:

- Describe how OEG manages certificates and keys
- Describe SSL support in the Enterprise Gateway
- Set up SSL
- Set up mutual SSL



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 10 Overview: Configuring SSL

This practice covers the following topics:

- Creating the certificate authority (CA)
- Creating the server certificate
- Configuring an HTTPS listener
- Setting up mutual SSL (optional)



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

11

Securing XML Messages

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Use policy to verify a digital signature
- Use policy to encrypt data
- Transform a message by using XSLT to remove sensitive data



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

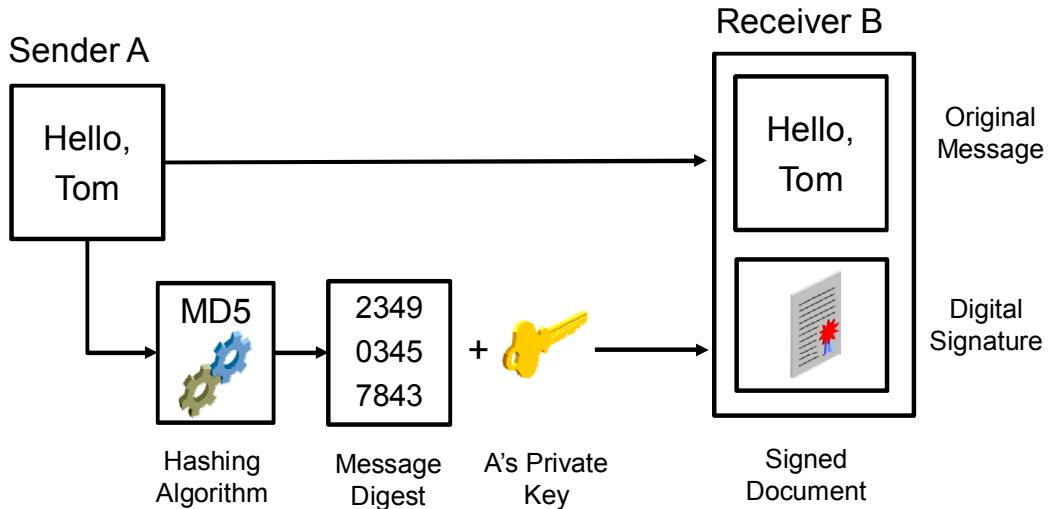
Roadmap

- XML signature
- XML encryption
- XML message transformation



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Digital Signature



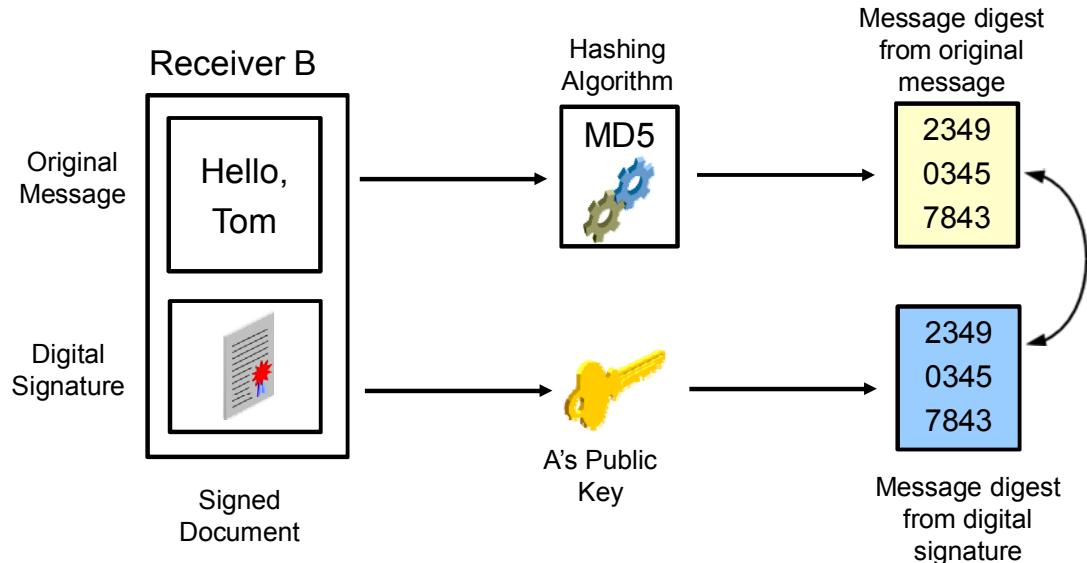
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The example shows a signing using asymmetric signature. With an asymmetric signature, the signatory's private key (from a public-private key pair) is used to sign the message. The corresponding public key is then used to verify the signature.

In the example, the message digest encrypted with the sender's private key is the digital signature.

Digital Signature



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The recipient hashes the message and compares the resultant message digest with the one from the digital signature.

XML Signature

XML Signature:

- Is a specification for expressing a digital signature in XML format
- Is used for signing “any digital content”—not just XML
- Defines an XML syntax for the context of the signature
- May be:
 - Enveloped (located in source XML)
 - Enveloping (wraps around the source XML)
 - External (located in a separate document from the source)



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The hashing and asymmetric encryption technologies (as explained in the previous couple of slides) are used to make the actual digital signature. XML Signature is a specification for expressing a digital signature in XML format. So, XML Signature is still “a digital signature.” Note, there is no such thing as “an XML Signature.”

XML Signature is used for signing “any digital content.” Just as XML Encryption is not just for encrypting XML, XML Signature is not just for signing XML.

XML Signature may be:

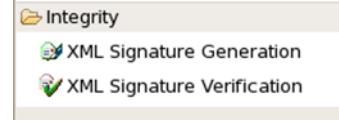
- **Enveloped:** XML signature located in source XML
- **Enveloping:** XML signature wraps around the source XML
- **External:** XML signature located in a separate document from the source

XML Signature Support in OEG

- An example of a signed XML message:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Header>
  <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse-
    <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" Id="Id-00013284649034
KZK7n0tX5LKhMMYCs2LtArBiKuFwU5JrVapgL3ZY301HTL7RKHImZn9bIXfCdDs
kk+LRPdkIfUhjDfIAAT2Pw02C3VGsJBm/uw3lFv08ZVpB8KGkl94Di7SvlFwiDeU
C3lCLZwxWAGJZmpU0RY1BLwTyMF29WIk1gNBk8UFk1Zu rmg60qj0z1i3SL8Dhdj
kAsCT5uGgyNc00sLlJh6pW0rpebwzXNkec0wu063A+RZCj6hjy/T8EwDtI+VjYP3
F4FnQ0BRr10TRfEK6Ry4xQ==</dsig:SignatureValue><dsig:KeyInfo Id="Id-0001328464903462-00000
  </wsse:Security>
</soap:Header>
<soap:Body>
  <ns:validateCard xmlns:ns="http://example.oracle.org/">
    <!-- Element must appear exactly once -->
    <ns:cardType>VISA</ns:cardType>
```

- XML Digital Signature has two filters:
 - One filter to generate
 - One filter to verify



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

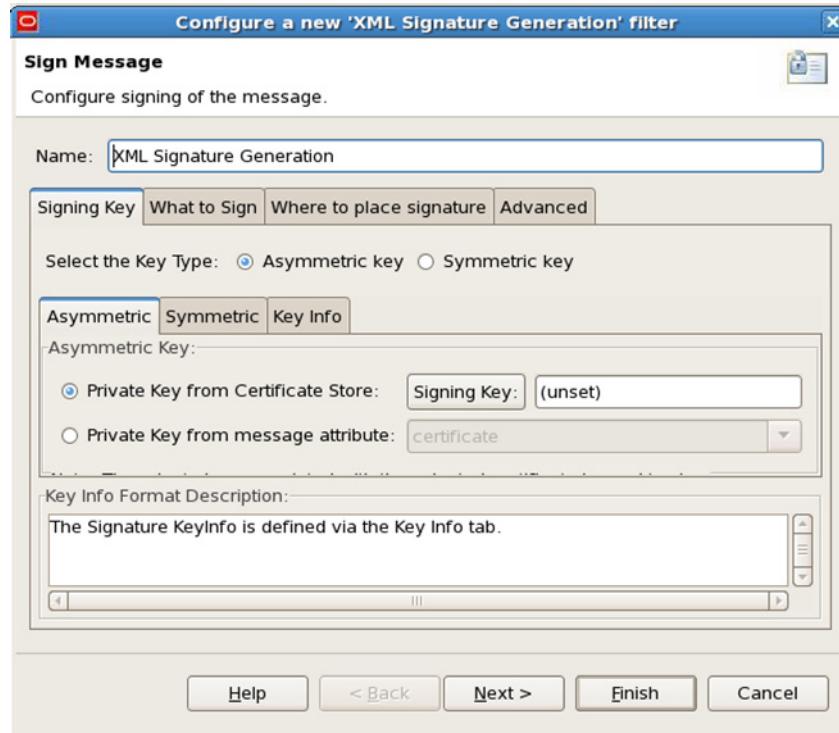
When signing an XML message, the resultant XML Signature is inserted into the message for consumption by a downstream web service. At the web service, the signature can be used to authenticate the message sender and/or verify the integrity of the message. In XML Signature, you can sign all or part of an XML document either by having the signature's Reference URI pointing internally or pointing to an external resource.

The Enterprise Gateway enables you to sign:

- SOAP messages
- Non-SOAP XML messages
- Message attachments

The Enterprise Gateway supports signing and verifying messages by using standards such as XML Signature and Secure Multipurpose Internet Mail Extension (SMIME). For XML Signature, Policy Studio provides two filters, XML Signature Generation and Signature Verification, under the Integrity category.

Generating XML Digital Signature



ORACLE®

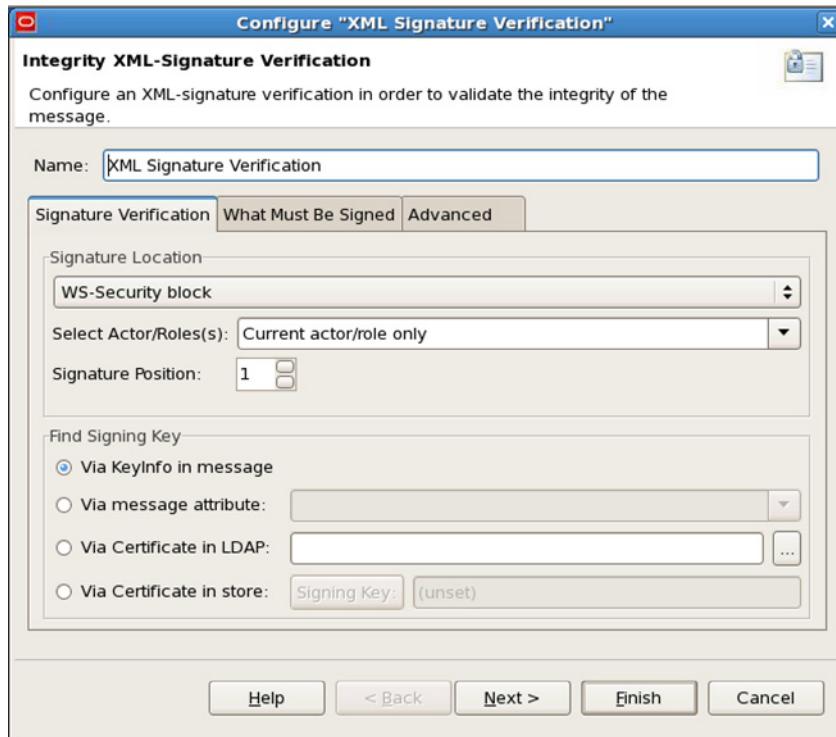
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can create an XML Signature Generation filter in order to set the settings for signing key, what to sign, what key to use, and so on.

- **Signature Key:** It is possible to use either a symmetric or an asymmetric key to sign the message content.
- **What to Sign:** Identify parts of the message that must be signed. Each signed part will be referenced from within the generated XML Signature. You can use any combination of Node Locations, XPaths, XPath Predicates, and the nodes contained in a Message Attribute to specify what must be signed.
- **Where to place the signature:** Specify where to insert the signature in the message. The supported options are:
 - Append Signature to Root or SOAP Header
 - Place in WS-Security Element for SOAP Actor/Role
 - Use XPath Location: This option is useful in cases where the signature must be inserted into a non-SOAP XML message.
- **Advanced:** Other advanced settings, such as specifying cryptographic algorithms and ciphers to sign the message parts; inserting a time stamp into the message to indicate when exactly the signature was generated

For more details about the settings, refer to the online help document.

Verifying XML Digital Signature



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In XML Digital Signature verification settings, you need to provide the following information:

- Where will the signature be located? A signature can be placed in the SOAP header, in a WS-Security block, or in a custom location in the incoming message.
- Where will the gateway find the certificate necessary to verify the signature? Typically, the certificate information comes from the KeyInfo block in the request. Alternatively, you can point to a specific certificate. In this case, the certificate comes in the KeyInfo block and you therefore need to configure the filter by selecting the “Find Signing Key: Via KeyInfo in message” option.
- What must be signed? Any part of the message can be specified. You can either use the predefined Node locations, or specify a custom location via XPath. For this sample, you need to specify a location, which is the Credit Card block.

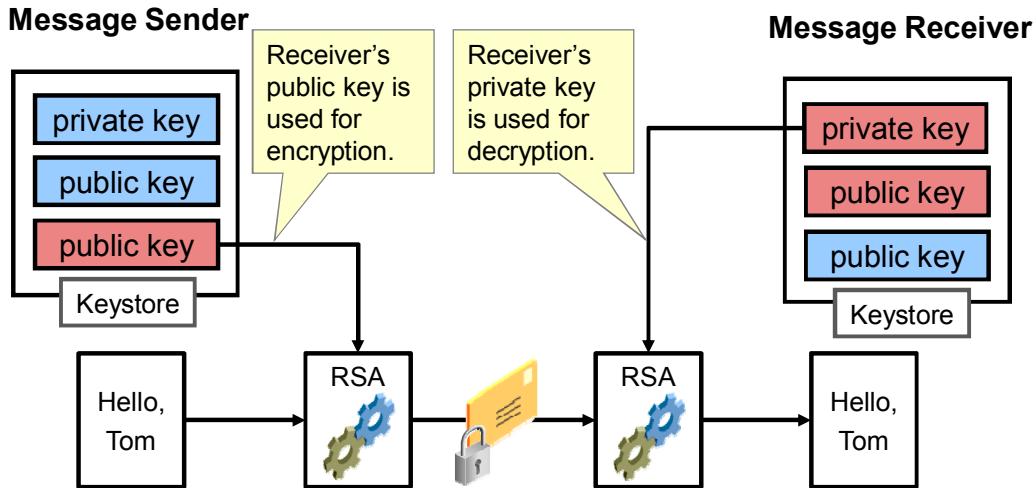
Roadmap

- XML signature
- XML encryption
- XML message transformation



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

XML Encryption



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

XML Encryption facilitates the secure transmission of XML documents between two application endpoints. Web services typically use the private key and digital certificate pairs for encrypting messages.

This diagram illustrates how messages are encrypted with the key pairs. As shown here, the sender and receiver both should have their own key pairs (public and private) and the public key of the other party. These keys are stored in the keystore.

When a sender sends an encrypted message, the sender uses the receiver's public key to encrypt the message so that only the designated receiver can use the appropriate private key to decrypt the message.

Messages can be encrypted symmetrically or asymmetrically. The diagram shows the encryption using the asymmetric keys.

Symmetric key encryption is very similar to SSL encryption: a symmetric key is used to encrypt the data itself and asymmetric encryption is used to carry the symmetric key within the message.

Basics of XML Encryption

XML encryption:

- Includes XML syntax for:
 - Representing encrypted XML content
 - Containing information needed to decrypt encrypted content
- Allows per-element encryption
- Tells the recipient how to decrypt the message



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Because XML Encryption followed XML Signature as a standard, the two share some of the same concepts, terminology, and XML elements. For example, you can sign all or part of an XML document in XML Signature. Similarly, in XML Encryption, you can encrypt all or part of an XML document either internally or externally. Another obvious similarity between the two is that they share the KeyInfo element, which was defined originally in the XML Signature namespace. However, XML Encryption addresses different issues than XML Signature.

XML Encryption enables you to encrypt any elements/nodes in a document, but the recipient may be able to decrypt only some of them.

XML Encryption: Example

Credit Card data to be encrypted:

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The example shows a SOAP message that has credit card information that needs to be encrypted.

XML Encryption Example

Encrypt the entire Credit Card XML block:

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <EncryptedData
    Type='http://www.w3.org/2001/04/xmlenc#Element'
    xmlns='http://www.w3.org/2001/04/xmlenc#'>
    <CipherData>
      <CipherValue>A23B45C56</CipherValue>
    </CipherData>
  </EncryptedData>
</PaymentInfo>
```

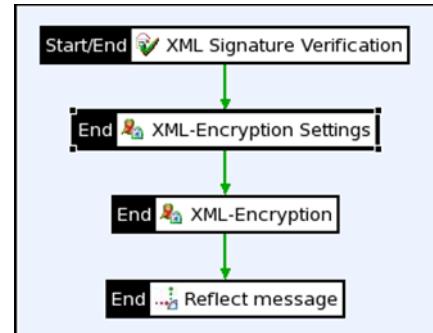


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

XML Signature points to what is being signed, whereas XML Encryption has the EncryptedData element, which typically contains the information that is being encrypted (in this example, the credit card information).

Creating Encryption Policy

- Get the recipient's public key:
 - Via Find Certificate
 - From successful Digital Signature Verification
- Use an "XML Encryption Settings" filter to configure what to sign.
- Use an "XML Encryption" filter in the path to execute the XML Encryption.



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

XML Signature and XML Encryption technologies are complementary because, with secure XML and web services, you need to both encrypt a message to a recipient as well as sign the message to confirm your identity and verify that the message you sent is the message that was received. The flow of the sample policy is:

1. After successful verification of the recipient's digital signature, the public key certificate is populated.
2. The gateway uses the recipient's public key to asymmetrically encrypt the symmetric key, which will then be transported to the recipient safely. The symmetric key, to be used to encrypt the data, is automatically generated by the gateway.

XML Encryption/Decryption Filter Settings



Encryption Settings include:

- Which key should be used
- What to encrypt (XPATH)
- Key Info
- Recipients
- Advanced: Algorithms mainly

Decryption Settings include:

- What to decrypt
- Decryption key



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When you encrypt or decrypt a message by using the filters, you typically need to configure the following fields:

- **EncryptedKey:** The EncryptedKey element encapsulates all information relevant to the encryption key.
- **EncryptionMethod:** The Algorithm attribute specifies the algorithm that is used to encrypt the data. The message data (EncryptedData) is encrypted by using the Advanced Encryption Standard (AES) symmetric cipher, but the session key (EncryptedKey) is encrypted with the RSA asymmetric algorithm.
- **CipherValue:** This is the value of the encrypted data. The contents of the CipherValue element are always Base64 encoded.
- **KeyInfo:** This contains information about the recipient and his/her encryption key, such as the key name, X.509 certificate, and Common Name.
- **EncryptedData:** This is the XML element(s) or content that has been encrypted. In this case, the SOAP Body element has been encrypted, and so the EncryptedData block has replaced the SOAP Body element.

Quiz

Which of the following statements are true concerning XML Signature and XML Encryption?

- a. Only SOAP messages can be signed and encrypted.
- b. All or part of an XML document can be signed and encrypted.
- c. XML Signature and XML Encryption are used to ensure that an XML document has not been tampered with.
- d. Both symmetric and asymmetric key can be used to sign and encrypt XML content.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b, d

Roadmap

- XML signature
- XML encryption
- XML message transformation

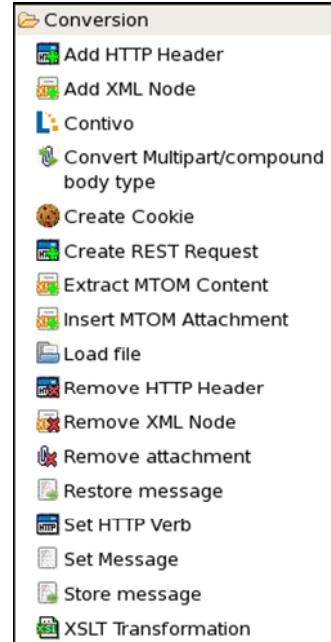


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

XML Message Transformation

OEG provides a number of ways to perform transformation:

- XSLT Transformation
- Add/remove HTTP Header
- Add/remove XML Node
- Set HTTP Verb
- Create REST Request
- Set/store message
- Convert Multipart/Compound Body Type
- Extract/insert MTOM Content
- Remove attachment
- Load File



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

OEG provides a number of ways to perform XML message transformation. You can find these filters under the Conversion category in Policy Studio. The commonly used filters include:

- **XSLT Transformation:** Convert the format of an XML message to another format by using an XSLT stylesheet.
- **Add HTTP Header:** Add an HTTP header to the message, and also set a Base64-encoded value for the header, if necessary.
- **Remove HTTP Header:** Remove a named HTTP header from the message to stop it from reaching the target system.
- **Add/remove XML Node:** Use this filter to add/remove an XML nodeset to the message.
- **Set HTTP Verb:** Set the HTTP verb (POST, GET, and so on) to use when routing messages to a web service.
- **Create REST Request:** Create HTTP requests to RESTful web services, and configure query string parameters sent with the REST request.
- **Set Message:** Replace the existing message body with alternative content.
- **Convert Multipart/Compound Body Type:** Compress a multipart (MIME) message to a ZIP file so that it can be archived.

- **Extract MTOM Content:** Extract the Base64-encoded contents from an element and create a MIME part with it according to the MTOM (Message Transmission Optimization Mechanism) specification.
- **Insert MTOM:** Insert the Base64-encoded contents of a MIME part into an XML message by using MTOM.
- **Remove Attachment:** Remove all attachments from an XML message.
- **Load File:** Load the contents of a specified file, and set it as message content to be processed.

XSLT

- An XSLT is used to transform an XML document into another document type.
- The stylesheet defines how elements in the XML source document should appear in the resulting XML document.
- XSLT is a template-based language.

For example, transform Celsius data to Fahrenheit data.

```
<xsl:template match="Temperature">
  <xsl:text>Fahrenheit =
    <xsl:value-of select="@value * 9 div 5 + 32"/>
  </xsl:text>
  <xsl:apply-templates/>
</xsl:template>
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Enterprise Gateway can transform an XML document into another document type by using the XSLT stylesheet. For example, an incoming XML message adhering to a specific XML schema can be converted to an XML message adhering to a different schema before it is sent to the destination web service.

This type of conversion is especially valuable in the Web Services arena, where a web service might receive SOAP requests from various types of clients, such as browsers, applications, and mobile phones. Each client might send up a different type of SOAP request to the web service. Using stylesheets, the Enterprise Gateway can then convert each type of request to the same format. The requests can then be processed in the same fashion.

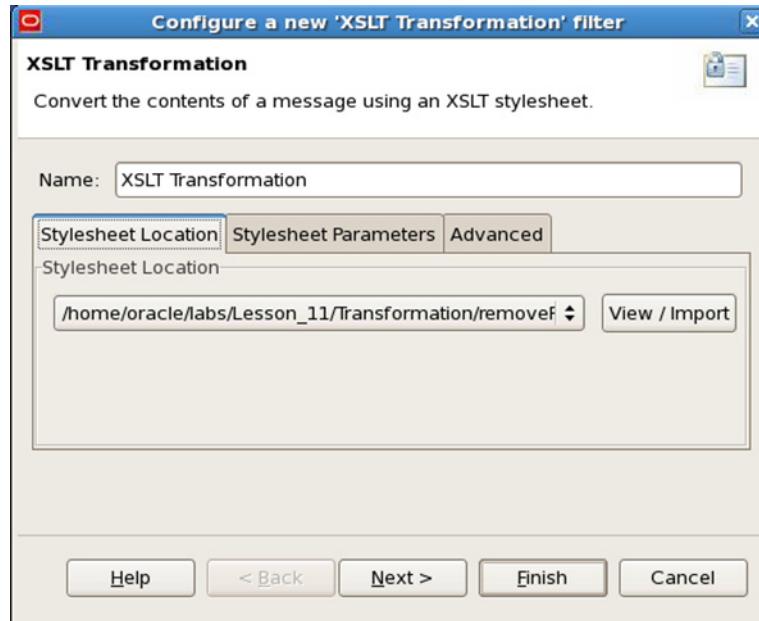
XSLT is a template-based language, as shown in the example:

```
<Temperature scale="Celsius" value="12"></Temperature>
```

becomes:

Fahrenheit = 53.6

XSLT Transformation Filter



ORACLE®

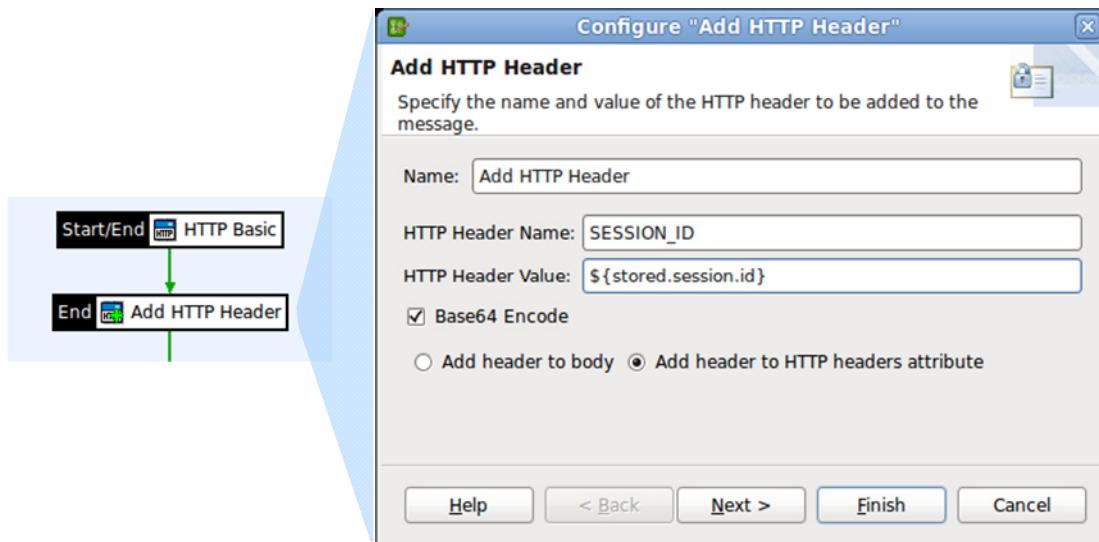
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In XSLT Transformation Filter, you need to configure the fields on the following tabs:

- **Stylesheet Location:** You can select an XSL stylesheet from the Stylesheet Location drop-down list, which is populated with the contents of the Stylesheet Library. You can import a new stylesheet into the library by clicking the View/Import button.
- **Stylesheet Parameters:** Using the XSLT Transformation filter, you can pass the values of message attributes to the configured stylesheet. For example, you can take the value of the `authentication.subject.id` message attribute, pass it to the configured XSL stylesheet, and then output this value to the result produced by the conversion.

Manipulating HTTP Headers

The gateway can add HTTP headers to a message as it passes through a policy.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The gateway can add HTTP headers to a message as it passes through a policy. The example shows that it is also possible to set a (Base64-encoded) value for the header.

Quiz

Choose the filters you can use to remove sensitive data from XML documents:

- a. Remove XML Node
- b. Remove HTTP Header
- c. Set HTTP Verb
- d. XSLT Transformation
- e. Remove attachment



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, d

Summary

In this lesson, you should have learned how to:

- Use policy to verify a digital signature
- Use policy to encrypt data
- Transform a message by using XSLT to remove sensitive data



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 11 Overview: Securing XML Messages

This practice covers the following topics:

- Creating Client Certificate and Key
- Verifying XML Signature
- Signing, encrypting data
- Transforming a message to remove sensitive data



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

12

Securing Web Services

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Use the WS-Security Username Token to authenticate a user
- Configure security policies from WSDL files



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

WS-Security: Overview

- Specifies rules to ensure:
 - Authentication—using security tokens
 - Confidentiality—using XML Encryption specification
 - Integrity—using XML Signature specification
- Supports multiple security tokens for authentication
 - Username/password
 - X.509 certificate
 - Kerberos ticket
 - Security Assertion Markup Language (SAML)
- Defines elements for packaging security tokens into SOAP messages



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Web Services Security (WS-Security) specification specifies the web services security at message level. IBM and Microsoft defined the specification, which is OASIS approved, “Web Services Security: SOAP Message Security”.

WS-Security is a collection of protocols that specify how different levels of security can be enforced on messaging in SOAP-based web services. It is meant to provide comprehensive end-to-end message content security and not just transport-level security. The security matters are not delegated to the transport level but are handled directly through an appropriate security API.

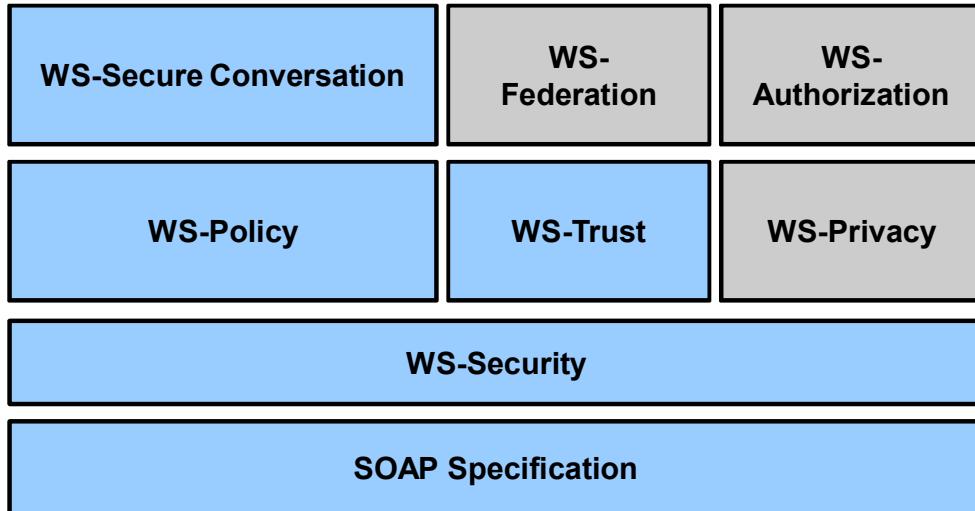
WS-Security defines SOAP extensions to implement client authentication, data integrity, and data confidentiality on the message level.

- Data confidentiality is to make sure that the data cannot be read during transit, by means of message encryption. WS-Security uses the XML Encryption specification to encrypt portions of the SOAP messages. Any portions of SOAP messages, including headers and body blocks, may be encrypted.
- Data integrity assures the recipient that the data that the application receives has not been altered during transit. It is implemented by XML Signature in WS-Security. XML Signature binds the sender’s identity (or “signing entity”) to an XML document. Signing and signature verification can be done by using asymmetric or symmetric keys.

- Authentication can be done by using security tokens. WS-Security enables you to use any security token you choose to use. Three different options are explicitly defined:
 - Username/password authentication in case of custom authentication
 - Binary authentication tokens in the form of Kerberos tickets or X.509 certificates
 - Security Assertion Markup Language (SAML)

WS-Security defines practically no new security technology; instead, it focuses on applying existing security technologies such as X.509 certificates, XML Signature, XML Encryption, and SAML assertions to SOAP messages.

WS-Security Stack



Not supported in WebLogic Server and not adopted by the industry



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

As shown in the slide, WS-Security is a family of specifications designed to augment transport-level security by providing a unified, transport-neutral, end-to-end framework for higher levels of security such as authorization. There are two layers above WS-Security: the first layer consists of WS-Policy, WS-Trust, and WS-Privacy; the second layer builds on the first layer and consists of WS-Secure Conversation, WS-Federation, and WS-Authorization.

First Layer

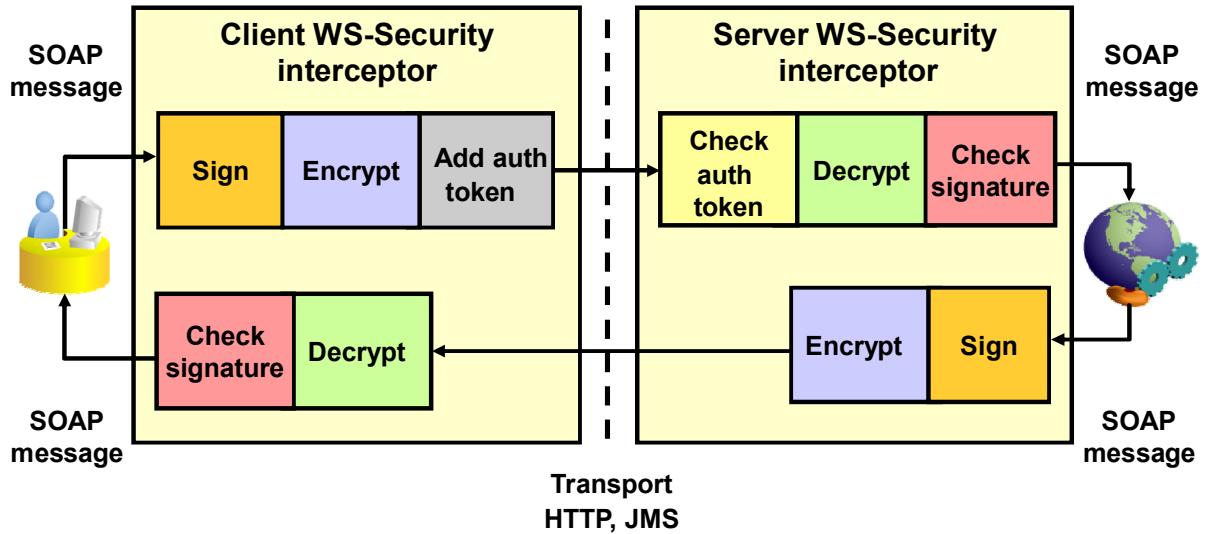
- **WS-Policy:** Describes general security capabilities, constraints, and policies. For example, a WS-Policy assertion could stipulate that a message requires security tokens or that a particular encryption algorithm should be used.
- **WS-Trust:** Deals primarily with how security tokens are to be issued and exchanged
- **WS-Privacy:** Explains how services can enforce privacy policies. The specification also covers how a service can determine whether a requester intends to follow such policies.

Second Layer

- **WS-Secure Conversation:** Secures web service conversations across different sites and, therefore, across different security contexts and trust domains. The specification focuses on how a security context is created and how security keys are derived and exchanged.

- **WS-Federation:** Addresses the challenge of managing security identities in a heterogeneous security environment. It describes how to maintain a single, authenticated identity across different platforms and organizations.
- **WS-Authorization:** Defines how web services manage authorization data and policies

WS-Security Architecture



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The slide displays the WS-Security architecture, which is in two parts: client WS-Security interceptor and server WS-Security interceptor. An interceptor is a prebuilt handler that implements all the WS-Security features, such as digital signatures, encryption, and authentication.

Whenever a client has to send a SOAP request message to the server, the SOAP request message is intercepted by the client interceptor. The interceptor adds the authentication, signature, and encryption elements to the SOAP message, and then forwards the message to the web service.

On receiving the message from the client, the web service invokes the server interceptor that verifies the signature of the incoming SOAP message. If the signature is valid, the interceptor decrypts and authenticates the message and forwards it to the web service endpoint implementation. After processing the client's request, the web service generates a SOAP response message that includes the WS-Security header with integrity and confidentiality information. The message is then sent back to the client.

On receiving the SOAP response message from the server, the client interceptor interprets the header and delivers it to the client application.

Basics of WS-Security

- WS-Security defines a Security element in the SOAP Header.

```
<S:Envelope>
  <S:Header>
    ...
    <Security S:actor="http://www.oracle.com/appml/"
      S:mustUnderstand="1">... </Security>
    ...
  </S:Header>
  ...
</S:Envelope>
```

- The WS-Security header contains time stamp, encryption, security tokens, and digital signature data.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

One simple and important thing that WS-Security does is to define a Security element to go into the SOAP Header. A SOAP message may contain multiple <Security> blocks, each for different actors (SOAP 1.1) or roles (SOAP 1.2).

WS-Username Token: Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header>
  ...
  <wsse:Security soap:actor="oracle"
    xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
    <wsse:UsernameToken wsu:Id="oracle"
      xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
      <wsse:Username>oracle</wsse:Username>
      <wsse:Password Type="wsse:PasswordText">oracle</wsse:Password>
      <wsu:Created>2009-05-19T08:46:04Z</wsu:Created>
    </wsse:UsernameToken>
  </wsse:Security>
</soap:Header>
<soap:Body>
  <getHello xmlns="http://www.oracle.com"/>
</soap:Body>
</soap:Envelope>
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The sample SOAP message in the slide shows the basic UsernameToken from the WS-Security specification. The client username and password are encapsulated in a WS-Security <wsse:UsernameToken>. You will notice that the UsernameToken uses a clear-text password. If you choose to use such an approach, you should at least consider running under SSL so that the message will be encrypted during transmission.

WS-Security provides an alternative to the clear-text password approach, which is UsernameToken with PasswordDigest. This approach is viable as long as you already have the clear-text password on both sides of the exchange. The process involves hashing some random information with the password and then sending it in the UsernameToken. To use the PasswordDigest type of UsernameToken, you hash together three items: a time stamp, a nonce, and the password itself.

WS-Security UserNameToken Support in OEG

- Both the gateway and Service Explorer enable you to:
 - Create WS-UserName tokens
 - Validate WS-UserName tokens
- Creation
 - Can generate basic or digest passwords
 - Can generate nonce and time stamps
- Validation
 - Uses nonce and time stamps to prevent replay attacks
 - Validate UserName or X509 DN against large set of user repositories (LDAP, internal User Store, Radius, Oracle Access Manager, CA SiteMinder, a database, and so on)



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

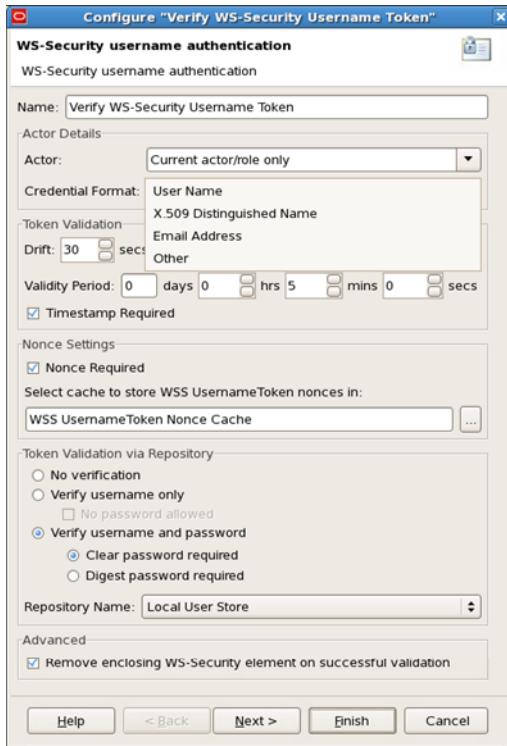
WS-Security specification supports security tokens such as username/password, X.509 certificate, or SAML assertion. These security tokens included in the message are typically used for authentication or authorization purposes.

A WS-Security Username Token enables an end-user identity to be passed over multiple hops before reaching the destination web service. The user identity is inserted into the message and is available for processing at each hop on its path.

When the Enterprise Gateway receives a Username token, it can perform one of the following tasks, depending on the requirements:

- Ensure that the time stamp on the token is still valid
- Authenticate the username against a repository
- Authenticate the username and password against a repository

Validating WS-UserName Token



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This slide shows the configuration that needs to be made in the Enterprise Gateway to authenticate users by using a WS-Security <wsse:UsernameToken>. The filter used is “WS-Security UserName Authentication” under the Authentication category.

For token validation, you should do the following:

- **Choose Role/Actor** (where the token is located): Specify which block contains the <wsse:UsernameToken> used to authenticate the end user.
- **Choose Credential Format:** The Enterprise Gateway can authenticate users against a user profile repository based on UserNames, X.509 Distinguished Names, or email addresses. Unfortunately, the WS-Security specification does not provide a means of specifying the type of <wsse:UsernameToken>, and so it is necessary for the administrator to do so by using the Credential Format field. The type specified here is used internally by the Enterprise Gateway in subsequent authorization filters.
- **Decide whether Time Stamp/Nonce are required** (Time stamp = Created node): Each wsse:UsernameToken contains a time stamp inserted into the <wsu:Created> element. Using this time stamp with the details entered in this section, the Enterprise Gateway can determine whether the WS-Security UsernameToken has expired.

- **Choose Validation Period** (including Drift): Drift time is specified in seconds to account for differences in the clock times between the machine on which the token was generated and the machine running the Enterprise Gateway.

Inserting WS-UserName Token



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When a client has been successfully authenticated, the Enterprise Gateway can insert a WS-Security Username Token into the downstream message as proof of the authentication event. The <wsse:UsernameToken> token enables a user's identity to be inserted into the XML message so that it can be propagated over a chain of web services.

You can configure the fields as follows:

- Get credentials (either hard-coded or from message attributes).
- Specify whether clear-text or digest password should be used.
- Optionally, indent the resulting XML.

Note that when using a digest password, only the digest is passed in the token. Validating a digest password is possible only against a user repository that lets the gateway get the password, calculate its digest, and compare it against what is in the token; but is not possible for LDAP, for example.

Quiz

OEG supports WS-Security Username Token with both clear-text password and digest password.

- a. True
- b. False



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a

WS-Policy: Overview

WS-Policy:

- Defines a framework for allowing web services to express their constraints and requirements
- Provides a model and the syntax for describing the policies of a web service
- Is divided into subsidiary specifications:
 - WS-Policy: Defines a grammar that explains web service policies
 - WS-PolicyAttachment: Associates policies with web services
 - WS-PolicyAssertions: Defines a set of general policy assertions



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Web Services Policy Framework (WS-Policy) is used by web services primarily to specify requirements that clients must satisfy to use the web service. The requirements cover security, encryption, transportation protocol, and so on. Basically, WS-Policy gives the web service an ability to specify requirements beyond what is normally definable by its WSDL specification. A policy consists of a set of rules written in XML that are either incorporated directly into or referenced by the web service's WSDL specification.

WS-Policy provides a model and syntax for describing the policies of a web service. It includes the following specifications:

- **WS-Policy:** Defines a grammar that explains web services policies
- **WS-PolicyAttachment:** Defines a policy regarding using attachments
- **WS-PolicyAssertions:** Defines a set of general policy assertions
Note: The constraints and requirements of web services are expressed as policy assertions.
- **WS-SecurityPolicy:** Defines a set of security policy assertions for use with WS-Policy to describe how messages are secured

Policy Assertion

- Policy assertion:
 - Is a basic unit representing individual requirement in a policy
 - Is domain specific (security, reliability)
- Service providers use a policy assertion to convey a condition under which they offer a web service.
- Example of policy expression:

```
<Policy>
  <wsp:TextEncoding Encoding="iso-8859-5" />
</Policy>
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A policy assertion is a piece of service metadata representing an individual requirement, capability, or other property of a behavior. Assertions are domain specific (security, reliability, and transaction), and the details are specified in separate specifications. Service providers use a policy assertion to convey a condition under which they offer a web service. A policy-aware client can recognize policy assertions and engage these behaviors automatically.

The policy expression (XML form of the policy) in the example in the slide consists of a `Policy` main element and a child element, `wsp:TextEncoding`. Child elements of the `Policy` element are policy assertions. This example represents a specific policy assertion for text encoding. A policy-aware client can recognize this policy assertion and engage text encoding automatically.

Note: The prefix `wsp` is used here to denote the WS-Policy XML Namespace.

Sample of a Policy File

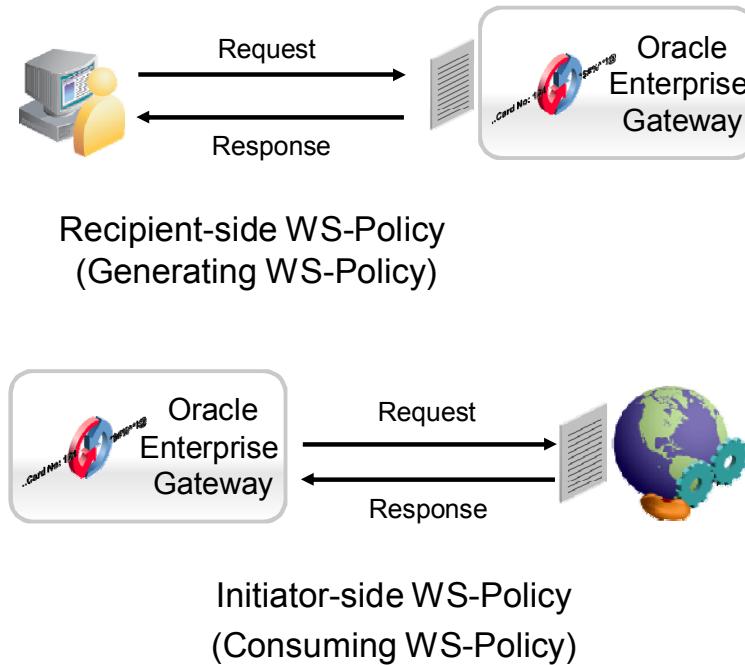
```
<wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
             xmlns:wsap="..." xmlns:ap="..." >
...
<wsp1_2:ExactlyOne>
  <wsp1_2:All>
    <ns2:SupportingTokens xmlns:ns2="http://docs.oasis-open.org/...">
      <wsp1_2:Policy>
        <wsp1_2:ExactlyOne>
          <wsp1_2:All>
            <ns2:UsernameToken>
              <wsp1_2:Policy>
                <wsp1_2:ExactlyOne>
                  <wsp1_2:All>
                    <ns2:HashPassword/>
                  </wsp1_2:All>
                </wsp1_2:ExactlyOne>
              </wsp1_2:Policy>
            </ns2:UsernameToken>
          </wsp1_2:All>
        ...
      </wsp1_2:Policy>
    ...
  </wsp1_2:All>
</wsp1_2:ExactlyOne>
</wsp1_2:Policy>
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This is an example of a WS-Policy file that specifies that a username token is required for authentication.

Securing a Service By Using WS-Policy



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When clients must use message-level or transport-level security mechanisms to communicate with the web service, you can include WS-Policy assertions in the WSDL, which are imported into the Web Services Repository to virtualize the back-end web service. These policy assertions can then be referenced at the operation or endpoint level in the WSDL. For example, a given web service may require the client to sign sensitive parts of the message and include a WS-Security UsernameToken to authenticate to the web service. You can include these policy requirements in the WSDL. Whenever a client retrieves the WSDL, it can automatically sign the relevant parts of the message and insert a valid UsernameToken.

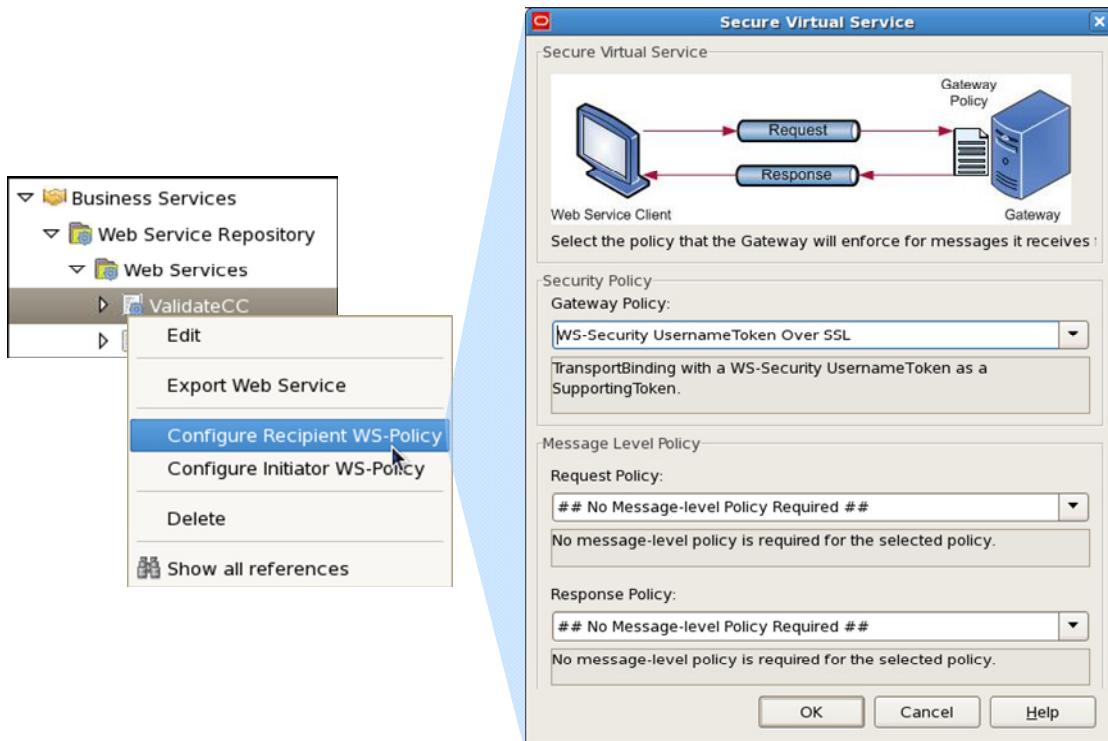
The Gateway can:

- Generate WS-Policies from a list of standard interoperability definitions and add those policies to the WSDL exposed by the gateway
- Consume WS-Policies attached to a service and generate the right policies to be executed before invoking that service

Recipient-side WS-Policy: The recipient WS-Policy defines the security contract between the client and the Enterprise Gateway. Any security tokens sent by the client are intended for consumption by the Enterprise Gateway. They are not intended for the back-end web service. On the recipient side, you will be asked about configuring inbound WS-Policy at service registration time.

Initiator-side WS-Policy: On the initiator side, the policy generation happens automatically when a WS-Policy is found in the WSDL (at service registration time).

Configuring Recipient WS-Policy



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When you import a WSDL file into the Web Services Repository, you can select the operations that you want to secure in the Import WSDL wizard. The Secure Virtual Service dialog box then enables you to specify the policy that the Enterprise Gateway enforces on the messages that it receives from the client.

In the Policy Configuration Settings wizard, you can then configure specific fields in the filters that are necessary to fulfill the security requirements specified in the Secure Virtual Service dialog.

Using policies in this way, the Policy Studio automatically generates the complicated circuits that the Enterprise Gateway uses to talk to the client. The Enterprise Gateway then becomes the recipient of the client, and is responsible for enforcing the selected policies on the messages that it receives from the client. The main advantage is that administrators can configure complex circuits to talk to clients in a secure manner with only a few clicks and minimal intervention.

Policy Configuration Settings



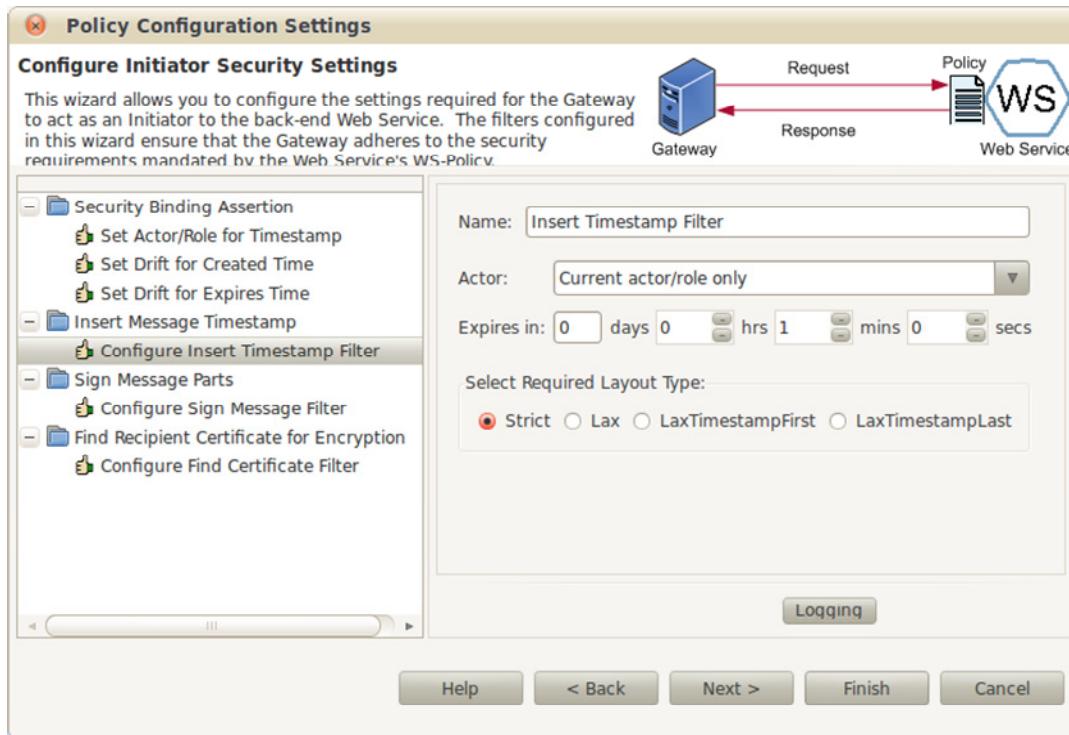
ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Depending on the policy configured in the Secure Virtual Service dialog box, the Policy Configuration Settings wizard displays configuration screens for the filters that implement the rules required by the configured policy. The exact sequence of screens differs depending on the policy that is selected.

For example, if a policy with a Username Token is selected, the Validate WS-Security UsernameToken filter is displayed. The effort in configuring these screens is minimal because the information is taken automatically from the WS-Policy assertions.

Configuring Initiator WS-Policy



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The filter will consume the WS-Policy included in the original imported WSDL. It happens automatically at service registration time.

Quiz

The _____ defines the security contract between the Enterprise Gateway and the back-end web service.

- a. Recipient WS-Policy
- b. Initiator WS-Policy
- c. None of the above



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

In this lesson, you should have learned how to:

- Use the WS-Security Username Token to authenticate a user
- Configure security policies from WSDL files



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 12 Overview: Securing Web Services

This practice covers the following topics:

- Authenticating the user by using WS-Security Username token
- Securing the communication between the client and the Enterprise Gateway by using WS-Policy



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

13

Securing SOA Composites with OEG and OWSM

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe security for SOA
- Explain the relationship between SOA Suite and Oracle Service Bus (OSB)
- Describe the OWSM architecture
- Describe how to use OWSM security policy to secure SOA composite applications
- Explain how to use OEG, OSB, and OWSM to provide end-to-end security for SOA composite applications



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

SOA and Web Services

Services are the SOA building blocks for a SOA application.

Web services is the most common way to implement SOA.

They:

- Address the low-level interactions between services
- Help up to a certain level of complexity in the infrastructure



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

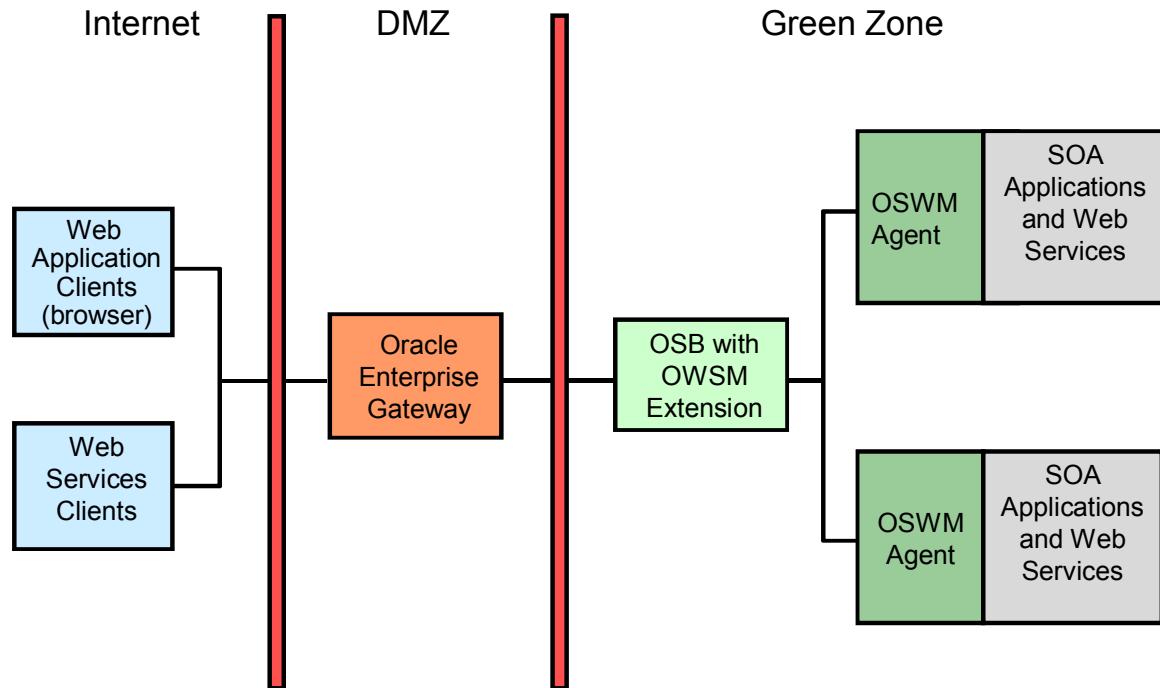
SOA is an approach to software design that involves assembling systems from reusable components or services that may have originated from different sources and underlying technology environments. SOA focuses on loosely coupled services, automated end-to-end business processes, and reusing existing IT systems resources based on open standards. At this level, SOA is often associated with Web services. However, Web services only address the low-level interactions between services. They do not address the architectural and design philosophy. SOA is more than just Web services. Most SOA is implemented by applying service-orientation principles to Web services technology.

You can think of SOA as an approach to service integration or, at a higher business level, an approach to application or system integration across organizational boundaries. Web services are essential, like building blocks that can be used (or reused) to enable a SOA application to be created to meet some business requirements.

Services are SOA building blocks. SOA applications can be thought of as functionality that aggregates a collection of related services, by reusing other services to complete automation of a business process.

Note: Web services are but one of the possible building blocks. An organization could build out an entire SOA by using, for example, Representational State Transfer (REST) services. Or, they may mix Web services and RESTful services, by using each where appropriate.

Securing SOA Composite Applications



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To protect SOA applications, it is recommended that you use a multilevel security architecture. In this deployment architecture, Oracle Enterprise Gateway components are placed in the demilitarized zone (DMZ). The connection between the client and the Enterprise Gateway is protected by a perimeter firewall.

Oracle Service Bus (OSB), Oracle Web Services Manager (OWSM) Agent, and Oracle Service-Oriented Architecture (SOA) are deployed in the green zone.

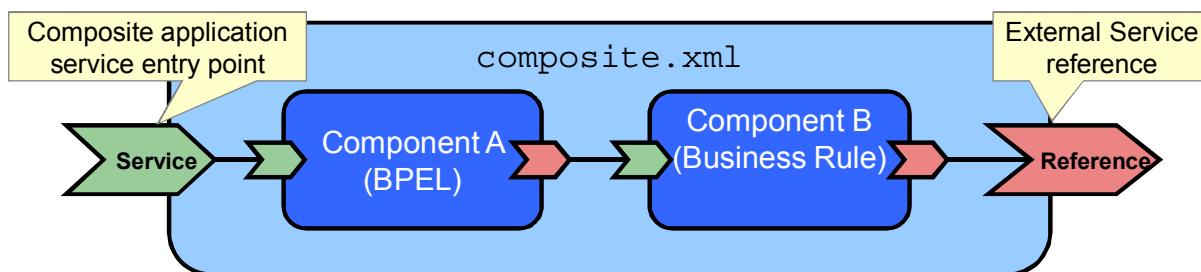
OSB is a configuration-based, policy-driven Enterprise Service Bus. It provides a robust shared-services layer that virtualizes the endpoints from the composite application, thus providing loose coupling between the composite and the applications.

OWSM is a security and management system that provides a common security infrastructure for web services and SOA applications.

SOA Composite Applications

SOA composite application is:

- An implementation of business functionality that manages information flow between varied sources
- A reusable functional unit that is:
 - Composed of several cooperating service components to meet a business process requirement
 - Deployed as a single composite application, including the component implementations



ORACLE

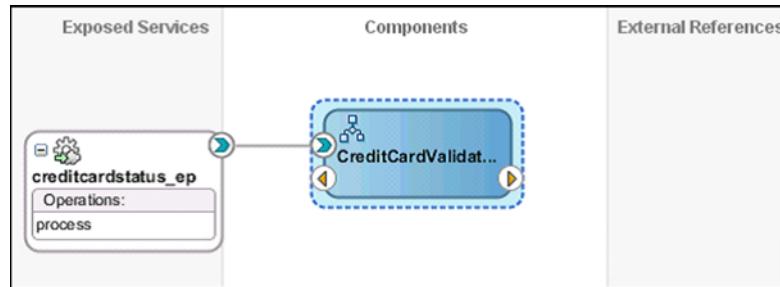
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Service Component Architecture (SCA) Applications

Service Component Architecture (SCA) is a set of specifications that prescribes the approach for assembling SOA composite applications (also known as the SCA Composite/Application or SOA Composite) from service components. The composite applications provide meaningful business functionality. The service components are pieces of business logic that are used as building blocks when composite applications are assembled. An SCA Composite may contain one or more cooperating component types such as BPEL process service components, Mediator components, and so on.

The SOA composite application is developed and deployed as a single service that includes all the components it assembles to form the application implementation. Components cannot exist on their own, or at least not in the SCA run-time environment. They need to be part of a service composite (application) because that is the unit of deployment and execution. When assembled together into a composite application, they are managed, maintained, and deployed together.

SOA Composite Application: Example



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The slide shows a simple example of SOA composite application, which contains only one service component named CreditCardValidationProcess. It is implemented in BPEL. `creditcardstatus_ep` is the service entry point that enables the composite accessible from outside the composite's domain.

Managing SOA Composite in Enterprise Manager

The screenshot shows the Oracle Enterprise Manager 11g Fusion Middleware Control interface. The left sidebar navigation includes Farm, SOA Infrastructure, Topology, Application Deployments, SOA, WebLogic Domain, Metadata Repositories, and User Messaging Service. The main content area is titled 'ValidationForCC [1.0]' and shows the 'SOA Composite' view. The top navigation bar includes Setup, Help, and Log Out. The dashboard header shows Running Instances 0, Total 5, Active, Retire..., Shut Down..., Test, Settings..., and a status message 'Page Refreshed Feb 1, 2012 1:13:40 AM UTC'. Below the header are tabs for Dashboard, Instances, Faults and Rejected Messages, Unit Tests, and Policies. The 'Component Metrics' section lists one component: CreditCardValida (BPEL), with 5 total instances, 0 running instances, and 0 faulted instances. The 'Services and References' section lists one service: creditcardstatus_ep (Web Service), with 0 faults and 0 total messages. The Oracle logo is visible at the bottom right.

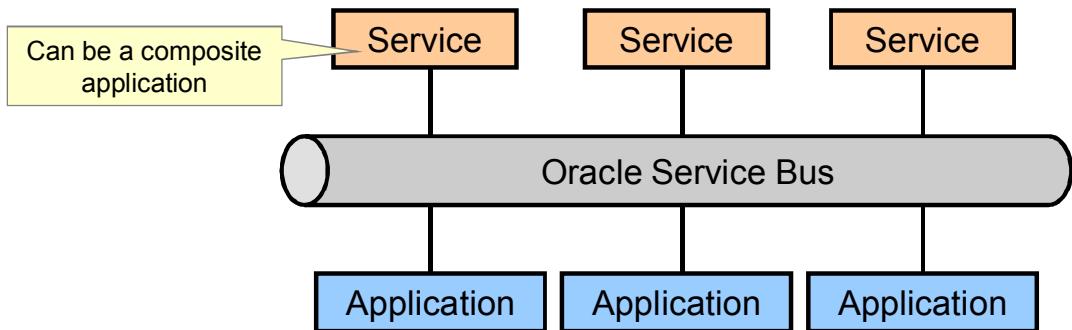
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE

SOA composite applications are deployed to the SOA infrastructure on a managed SOA server. Deployed composite applications are managed and can be tested in Oracle Enterprise Manager Fusion Middleware Control.

Oracle Service Bus

- Provides stand-alone service bus capabilities that support agility
- Provides scalability
- Insulates the integration logic
- Provides service use and tracking
- Provides service management



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

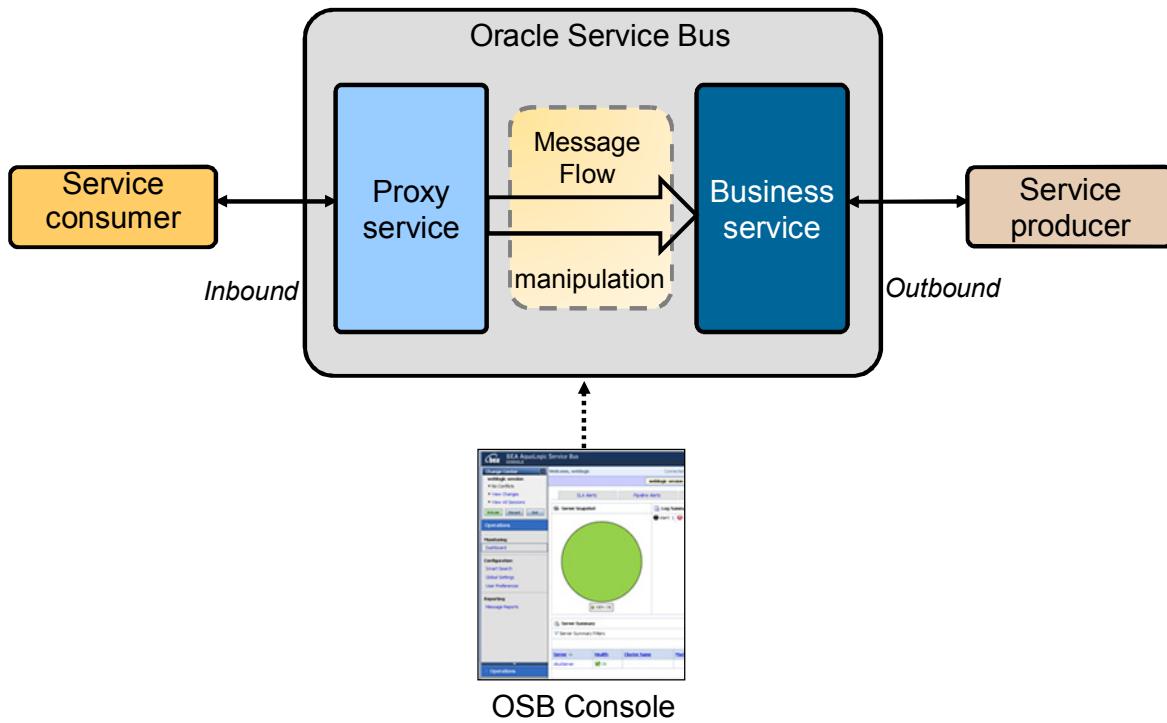
Oracle Service Bus (OSB) is a configuration-based, policy-driven Enterprise Service Bus (ESB). It provides a feature-rich console for dynamic service and policy configuration, as well as for system monitoring and operations tasks. OSB facilitates a loosely coupled architecture and an enterprisewide reuse of services, and centralizes management.

OSB:

- Filters, transforms, and routes messages with many different capabilities
- Insulates integration logic by creating virtual service interfaces for SOA composite applications and ESB implementations
- Provides service tracking and service management functionality

Note: OSB is a different product that is installed separately from Oracle SOA Suite 11g. OSB is a technology that is complementary to Oracle SOA Suite 11g and earlier releases of Oracle SOA products.

Oracle Service Bus Architecture



ORACLE®

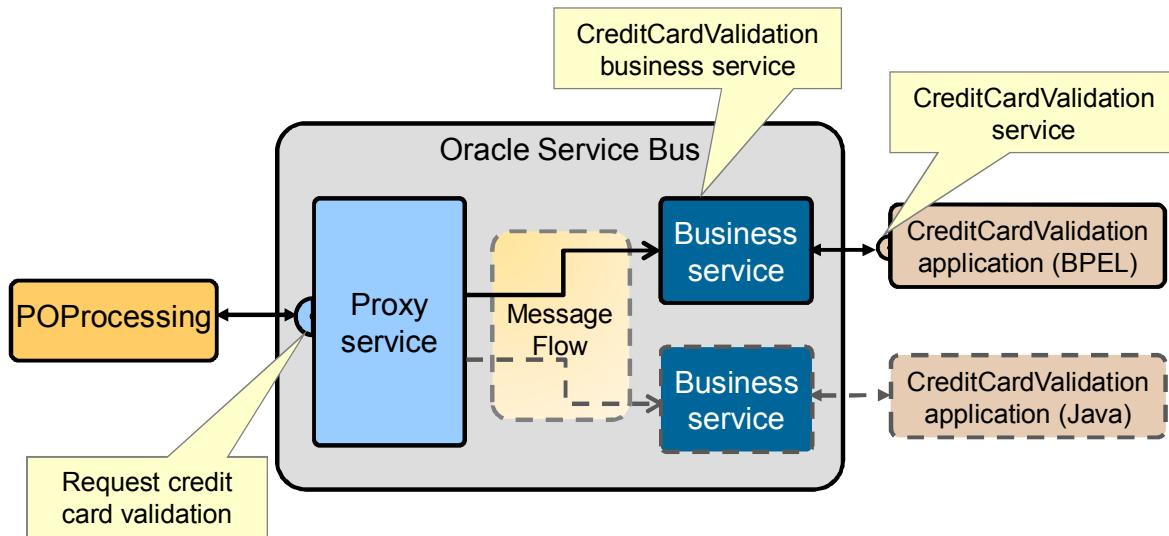
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Oracle Service Bus appears to its consumers as a service provider. What goes on inside OSB and how it interacts with the “real” services that do the actual work is hidden from consumers. The services exposed by OSB are called proxy services. Proxy services are definitions of generic intermediary web services that are hosted locally on OSB. Proxy services can route messages to multiple underlying services that do the actual work—called business services that use their configured independent interfaces. Proxy services can be defined and configured by using the OSB console. They are configured by specifying their interface, type of transport OSB uses, and their associated message processing logic.

Business services are OSB definitions of the enterprise services that exchange messages during business processes. A business service and its interface can be defined and configured by using the OSB console. A business service is configured by specifying its interface, the type of transport it uses, its security requirements, and other characteristics.

The graphic in the slide depicts OSB exposing the proxy services that connect through a message flow (including routing and transformation logic) to business services. The inbound transport layer is the communication layer between the client services (or service consumers) and OSB. The outbound transport layer is responsible for the communication between the service producers and OSB.

Virtualizing Service By Using OSB: Example



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The proxy service and business service architecture provide service virtualization. Through a proxy connection, OSB hides the actual producer services from the consumer. The virtualization adds decoupling between the producer and its consumers. This insulation layer can also help to shield consumers from changes in the underlying service.

You can see a simple use case here. Suppose you have a Order Processing composite application that invokes the CreditCardValidation service to validate the credit card number. Both the Order Processing composite and the CreditCardValidation service resided in the same service domain. However, the CreditCardValidation service could be provided by a credit card company (with its own business domain) and the company may not allow external parties to call into their services the same way that you did. With OSB, you can virtualize the CreditCardValidation service, decoupling it from the Order Processing service domain. External consumers calling into the CreditCardValidation service are mediated by OSB.

When the credit card company, at some point, migrates to a different application, it is able to continue offering the same validation service—with OSB then wrapping a completely different underlying application, for example, a Java web service.

OSB Console

The screenshot shows the Oracle Service Bus 11gR1 Dashboard - SLA Alerts page. The left sidebar has sections for Change Center, Operations (Monitoring, Configuration, Reporting), Resource Browser, Project Explorer, Security Configuration, and System Administration. The main content area has tabs for SLA Alerts, Pipeline Alerts, Service Health, and Server Health. The SLA Alerts tab is selected, showing a summary of SLA Alerts (30 mins) and Services With Most Alerts. Below this, there's an Alert History section with a 'Table Customizer' button highlighted by a red box. The table header includes columns for Timestamp, Alert Name, Alert Severity, Service Type, Action, and Service. Both alert history tables show 'No Alerts to display.' and 'Items 0-0 of 0'.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE

The OSB console is a web application deployed on a WebLogic Server instance that is used for two purposes:

- Managing and configuring your OSB environment, which is done at run time by administrators
- Developing and configuring services, which is done at design time by developers and architects

Note: Developers can also install Eclipse with the OSB Workshop plug-in for file-based service development.

To access the OSB console, enter the following URL in your browser:

`http://<hostname>:<port>/sbconsole`

where `<hostname>` represents the name of the machine on which WebLogic Server is running and `<port>` represents the port number. In this example, the URL is `http://localhost:7001/sbconsole`.

You can customize the console by using the Table Customizer to display table information according to your specifications. The default sort order for any table is determined by the first column in the table.

The different sections in the OSB console are briefly described as follows:

- **Operations:** The Operations module contains options to view and monitor OSB health and alerts.
- **Resource Browser:** The Resource Browser provides tools for locating resources and performing simple actions on them, such as Display, Edit, Export, and Delete.
- **Project Explorer:** The Project Explorer organizes configurations and resources of proxy and business services into projects and folders.
- **Security Configuration:** You use the Security Configuration module to create and modify security data that is used in Oracle Service Bus inbound security and administrative security.
- **System Administration:** The System Administration module provides tools for locating UDDI registries, and importing and exporting resources.

Quiz

When using OSB, a service consumer interacts with the:

- a. Service producer
- b. Proxy service
- c. Business service
- d. None of the above

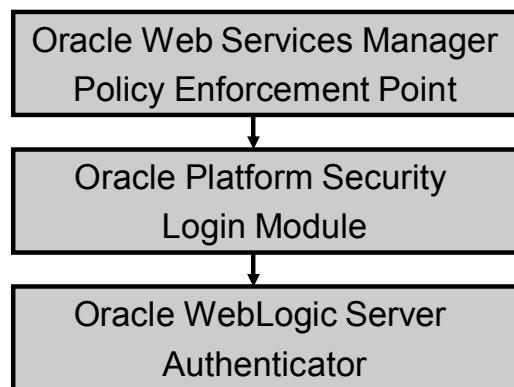


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Oracle Web Services Manager Policy Framework

- It provides a policy framework to manage and secure web services consistently.
- The policy framework is built by using the WS-Policy standard, and leverages the Oracle Platform Security Services (OPSS) Login Module and Oracle WebLogic Server authenticator for authentication and authorization.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

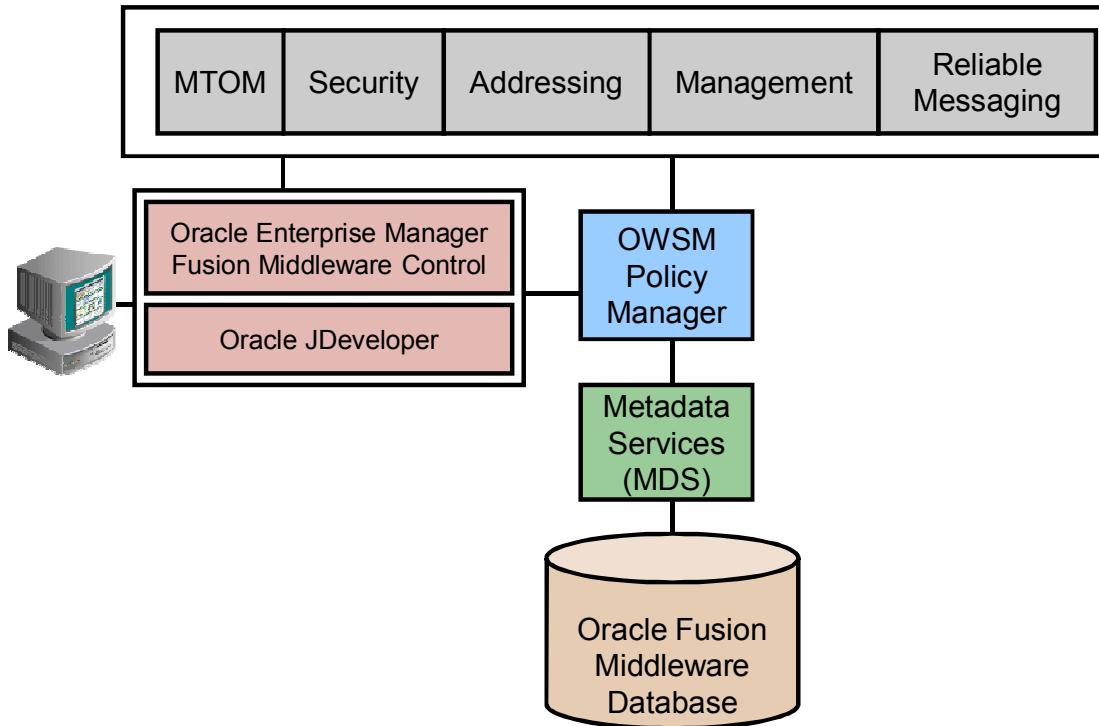
OWSM can be leveraged from Oracle Enterprise Manager Fusion Middleware Control to:

- Centrally define policies by using the OWSM Policy Manager
- Enforce OWSM security and management policies locally at run time

The tasks that can be performed from OWSM are:

- 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

Components of the Oracle Web Services Manager Architecture



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The components of the Oracle Web Services Manager architecture can be described as follows:

- **Oracle Enterprise Manager Fusion Middleware Control:** Enables administrators to access Oracle Web Services Manager's functionality to manage, secure, and monitor web services
- **Oracle Web Services Manager Policy Manager:** Reads/writes the policies, including predefined and custom policies from the Metadata Services (MDS)
- **Oracle WSM Agent:** Manages the enforcement of policies via the Policy Interceptor Pipeline
- **Policy Interceptors:** Enforce policies, including reliable messaging, management, addressing, security, and Message Transmission Optimization Mechanism (MTOM)
- **Metadata Services:** Used for storing 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

Supported Policies

The different types of policies supported in Oracle Fusion Middleware 11g R1 are:

- WS-ReliableMessaging
- Management
- WS-Addressing
- Security
- Message Transmission Optimization Mechanism (MTOM)



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

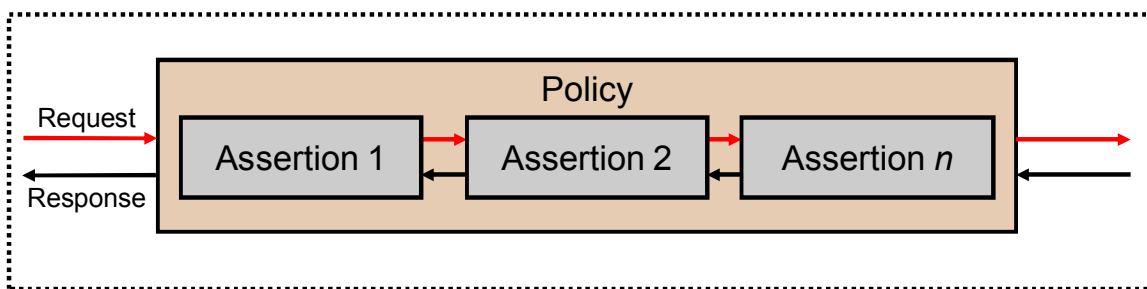
The different type of policies available are as follows:

- **WS-ReliableMessaging:** 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 sequential order in which a set of messages are delivered.
- **Management:** Management policies that log request, response, and fault messages to a message log. Management policies may include custom policies.
- **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.
- **Message Transmission Optimization Mechanism:** 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.

Oracle Web Services Manager Policy Assertions

- An OWSM policy consists of one or more assertions that exhibit a particular behavior.
- Assertions are executed in the order in which they are listed in the policy.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Oracle Web Services Manager (OWSM) policies consist of one or more assertions exhibiting a particular capability or behavior. A policy assertion is the smallest unit of a policy that performs a specific action. For example, a security policy could be made up of two assertions:

- A Log assertion
- A WS-Security assertion

If this particular security policy is attached to a service endpoint, then for the request message, the log assertion is executed first, logging the request message to a log file; then the WS-Security assertion that authenticates the requestor based on the token sent in the message decrypts the message (if the message is encrypted).

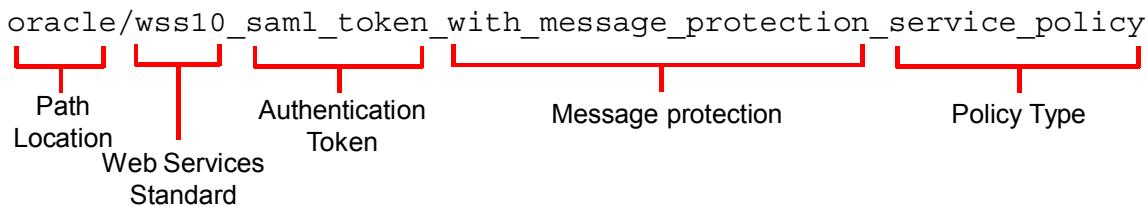
OWSM policy assertions are instances of policy assertion templates that are added to a policy at policy creation time. OWSM:

- Provides a set of predefined policy assertion templates
- Enables users to define custom policy assertions that can be combined with predefined policy assertions

Note: Custom policy assertions are used when specific functionality is not provided.

Oracle WSM Predefined Policies and Assertion Templates

- A set of predefined policies and assertion templates are available by default.
- You can directly attach these predefined policies to your web services or clients.
- The naming conventions for security policies:



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

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

You can attach these predefined policies to your web services or clients, and configure them, or create a new policy by making a copy of one of the predefined policies.

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

Applying Policies to Service in EM

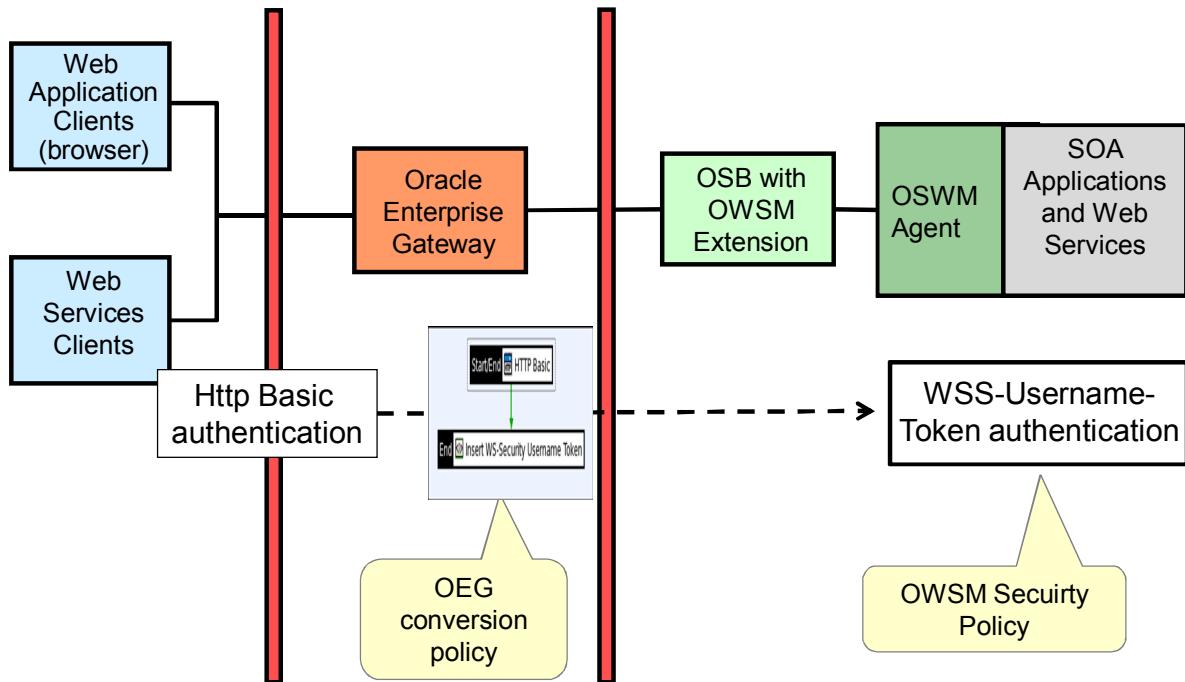
The screenshot shows the Oracle Enterprise Manager 11g Fusion Middleware Control interface. The left sidebar shows the navigation tree with 'Farm_base_domain' selected. Under 'Farm_base_domain', 'SOA' is expanded, showing 'soa-infra (soa_server1)' and 'default'. 'ValidationForCC [1.0]' is selected under 'default'. The main content area displays the 'ValidationForCC [1.0]' application details. The 'Policies' tab is active. A message states: 'You can view and manage the list of policies attached to the web service bindings and components of this SOA composite application. Click 'Attach To/Detach From' to update the list of attached policies.' Below this, a table lists policies attached to the service endpoint 'creditcardstatus_ep'. The table has columns: Policy Name, Attached To, Policy Reference Status, Category, and Tc. Two policies are listed: 'CreditCardValidationProcess' and 'creditcardstatus_ep'. The 'creditcardstatus_ep' row is highlighted with a blue background.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can attach policies to a service endpoint by using the Oracle Enterprise Manager Fusion Middleware Control console at run time.

Converting Authentication with OEG Policy

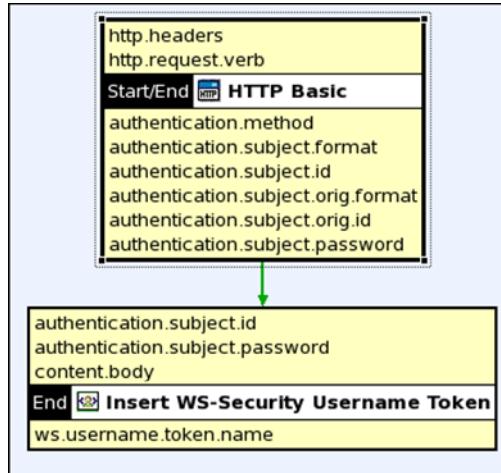


ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

If clients use different authentication, in this case, HTTP Basic, a conversion policy needs to be created, and applied to the web service registered in OEG.

The Conversion Policy



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

With this policy, a user is authenticated to the Enterprise Gateway using HTTP Basic Authentication. After successfully authenticating the user, the Enterprise Gateway inserts a WS-Security Username Token into the downstream message. The WS-Username Token enables a user's identity to be inserted into the XML message so that it can be propagated over a chain of web services.

Quiz

You can attach OWSM policies to a service endpoint by using:

- a. OEG Policy Studio
- b. OSB Console
- c. EM Console
- d. WebLogic Server Administration Console



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Summary

In this lesson, you should have learned how to:

- Describe security for SOA
- Explain the relationship between SOA Suite and Oracle Service Bus (OSB)
- Describe the OWSM architecture
- Describe how to use OWSM security policy to secure SOA composite applications
- Explain how to use OEG, OSB, and OWSM to provide end-to-end security for SOA composite applications



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 13 Overview: Securing SOA Application

This practice covers the following topics:

- Deploying and Examining the SOA application
- Virtualizing the Web Service in OSB
- Registering the Web Service in OEG
- Applying a OWSM Security Policy to the Web Service
- Adding a Authentication Conversion Policy to the Registered web service in OEG



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Integrating with Identity and Access Management

14

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

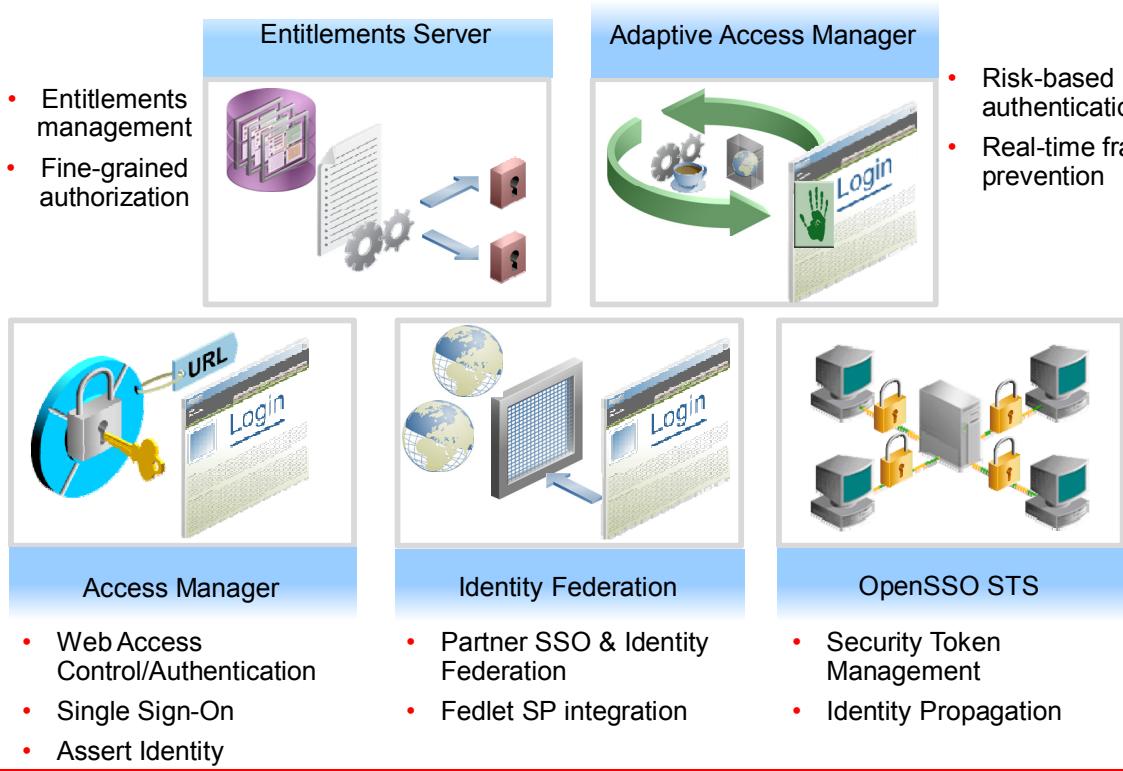
After completing this lesson, you should be able to:

- Describe how OEG works with Oracle Access Manager (OAM) for authentication
- Describe how OEG leverages Oracle Entitlements Server (OES) for fine-grained authorization



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Oracle Access Management Suite



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

OAM 11g is a Web SSO solution that is primarily focused on controlling access to resources that are accessible via HTTP, that is, resources identified by a URL.

OAM primarily performs three tasks:

- Providing authentication services
- Managing sessions and SSO for the Web tier
- Asserting identities to the application tier

OAM consists of two components: Agents and the OAM server.

Protected resources are the applications that are protected by OAM 11g (also referred to as “partner applications”). Access to these resources is subject to the access control policies in OAM 11g and is enforced by enforcement agents that are deployed in the access path of the protected resource (for example, Web agents deployed in the Web server and J2EE agents deployed in the application server).

Agents are entities that control access to protected applications based on security policies. Agents that are present in the resource access path intercept every resource access request to enforce the security policy that protects the resources. Because agents have to integrate with the protected application environment. The following agents are supported in OAM 11g R1:

- OAM 10g/11g WebGates
- Oracle Single Sign-On (OSSO) 10g Apache Module (mod_osso)
- OAM 10g AccessGates

A protected application must be protected by a WebGate or mod_osso instance that is registered with Oracle Access Manager as a policy enforcement agent. The policy enforcement agent acts as a filter for HTTP requests. Oracle Access Manager enables administrators to define authentication and authorization policies.

OAM Server:

This is the server-side component that provides core run-time access management services. OAM server is a J2EE application that provides various IdP (Identity Provider) services. The OAM server provides SSO, authentication, and authorization services (coarse level).

Oracle Entitlement Server:

- Provides authorization services
- Provides fine-grained application and data-level security
- Manages entitlements, grants, and policies

Oracle Adaptive Access Manager:

- Provides fraud detection services to the access suite
- Strengthens authentication security
- Enables transaction security

Oracle Identity Federation:

- Provides cross-domain SSO
- Manages trust across partners
- Enables interoperability across providers

Key Features of OAM

Oracle Access Manager provides:

- Authentication service for Web-based applications
- Single sign-on access for applications
- Identity assertion service
- Session management
- Coarse-level authorization protection



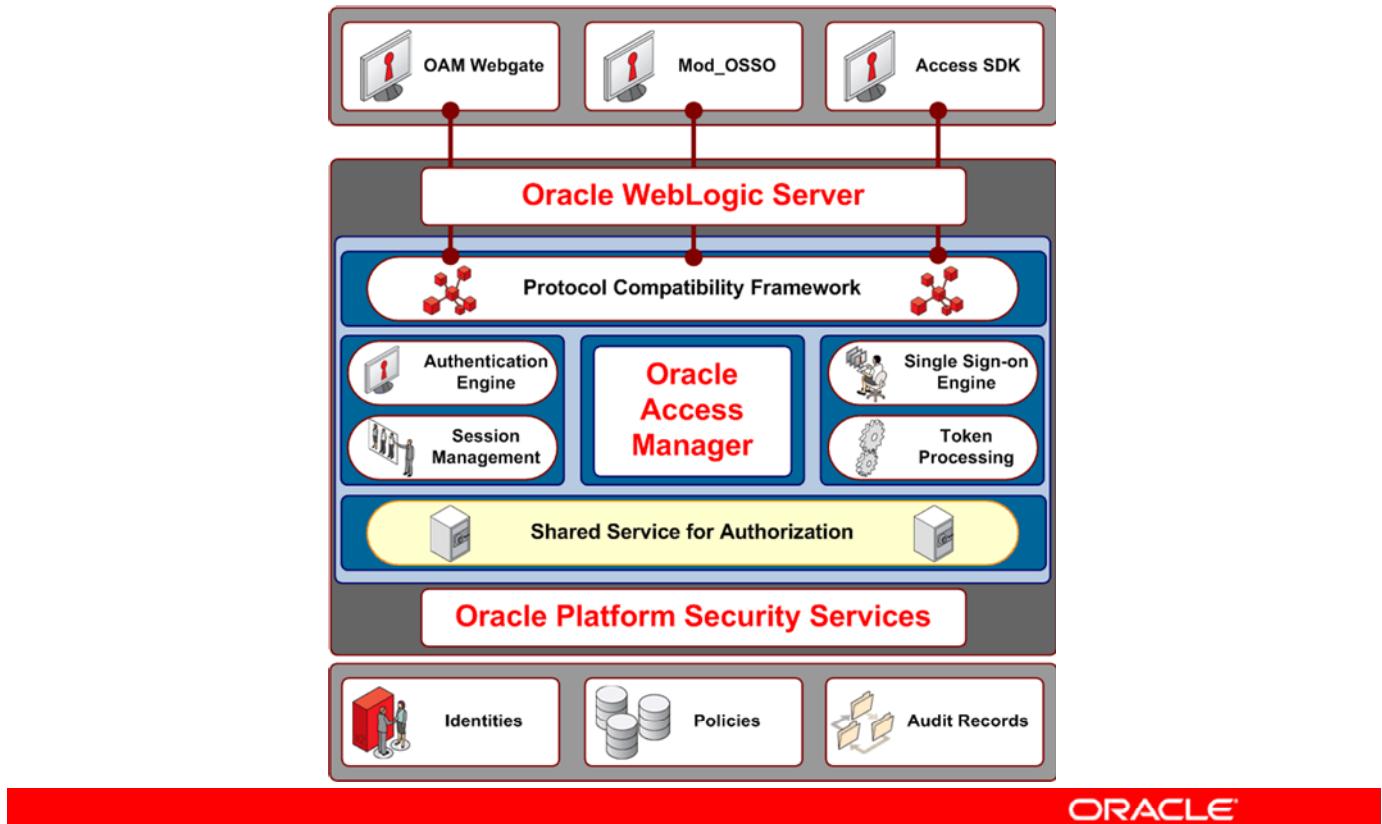
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Single sign-on is a process that provides users the ability to access multiple protected resources (Web pages and applications) with a single authentication. Oracle Access Manager 11g provides single sign-on (SSO) through a common SSO engine that provides consistent service across multiple protocols. Oracle Access Manager 11g is the Oracle Fusion Middleware 11g single sign-on solution. Single sign-on enables users and groups of users to access multiple applications after a single sign-on and authentication.

The goal for OAM 11g R1 is to provide a single authoritative source for all authentication and authorization services. The core service provided is the checking of valid session tokens, the requesting of credentials if the session token is invalid or missing and the issuing of session tokens, intercepting resource requests, and evaluating access control policies to control access to resources.

OAM accomplishes single sign-on by managing the user session life cycle, which also involves facilitating global logout by orchestrating logout across all relying parties in the valid user session. OAM also ensures that resource access by users is authorized, subject to the specified authorization policy.

OAM Architecture



ORACLE®

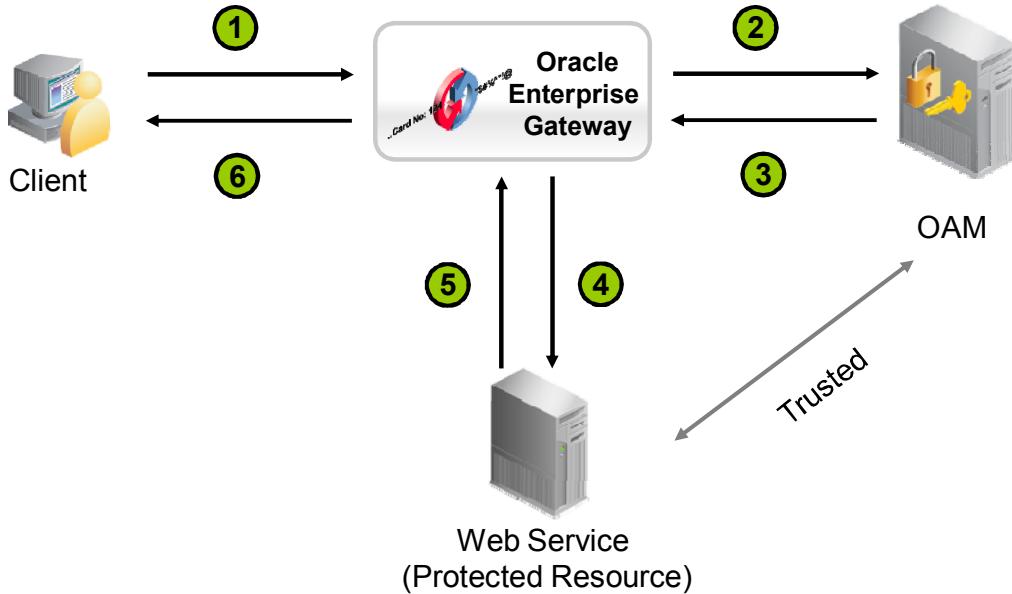
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The two primary types of actors in OAM architecture are the policy servers (OAM servers) and OAM policy enforcement agents (WebGates or AccessGates). In the security world, agents represent the policy enforcement point (PEP), whereas OAM servers represent the policy decision point (PDP).

The agent plays the role of a gatekeeper to secure resources such as HTTP-based applications, and manages all interactions with a user who is trying to access that resource. This is accomplished according to access control policies that are maintained on the policy server (OAM server).

The role of the OAM server is to provide policy, identity, and session services to the agent to properly secure application resources, authenticate and authorize users, and manage user sessions.

Authentication By Using OEG and OAM



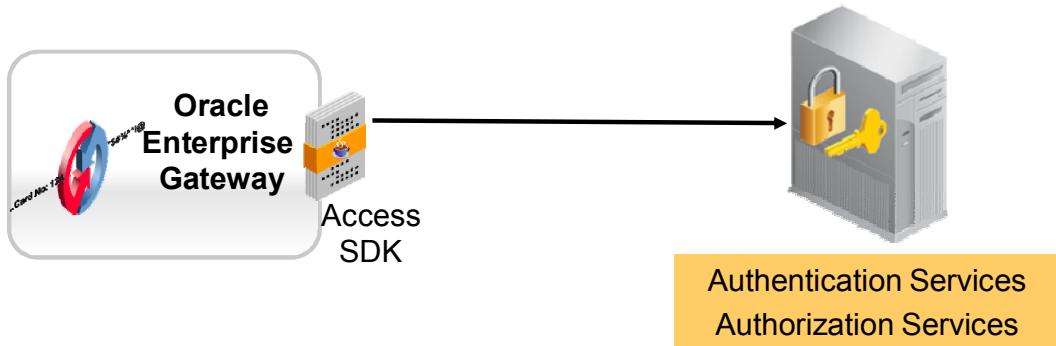
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A graphical representation of the flow of a message through OEG Gateway authenticating and authorizing a user via Oracle Access Manager and finally passing it onto the protected resource.

1. Request by using Basic HTTP Authentication.
2. Gateway connects to OAM for authentication and authorization.
3. OAM authenticates and authorizes the client.
4. Gateway sends request to protected resource.
5. Web service sends response to Gateway.
6. Gateway adds SSO token into response header, and sends response to gateway.

OEG Gateway as an AccessGate



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The OEG Gateway acts as an AccessGate 10g to Oracle Access Manager. AccessGates are Oracle Access Manager clients. They process requests for access to resources within the domain protected by your Access Manager. If a resource is not protected, the AccessGate grants the user free access to the requested resource. If the resource is protected and the user is authorized to provide certain credentials to gain access, the AccessGate attempts to retrieve those user credentials so that the Access Server can validate them. If authentication of the user and authorization for the resource succeed, the AccessGate makes the resource available to the user.

The Access SDK must be installed on the machine running the OEG Gateway so that the Gateway can act as an AccessGate.

Access SDK is a set of application programming interfaces (APIs) with which programmers can call Oracle Access Manager from programs written in the following languages:

- Java
- C
- C++
- C#

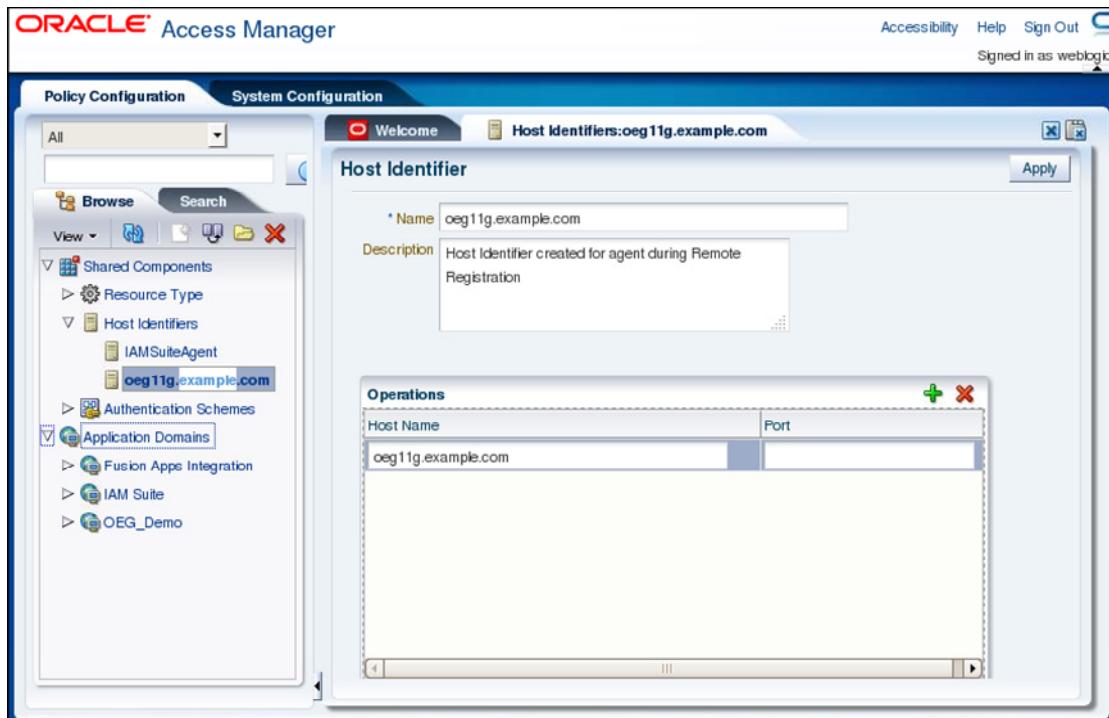
Key Steps of Authentication via OEG and OAM

- Install AccessSDK.
- Create an AccessGate entry on the OAM server.
- Configure Oracle Access Manager Authentication Repository in Policy Studio.
- Create Authentication policy by using Policy Studio.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Creating AccessGate Entry



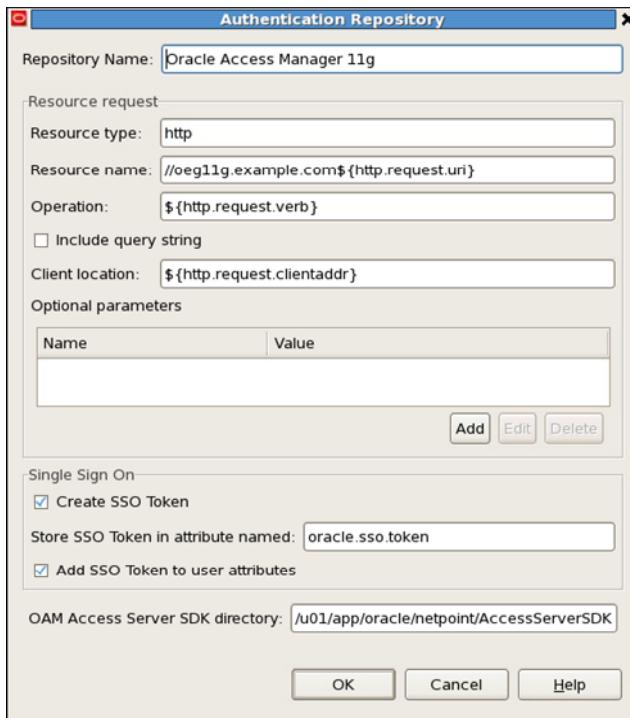
ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

After the Access Manager SDK has been installed, an AccessGate entry needs to be created on the Access Server.

During the creation, you need to provide “Host Identifier,” which should be the DNS name of the machine hosting the server instance on which the AccessGate resides.

Configure OAM Authentication Repository



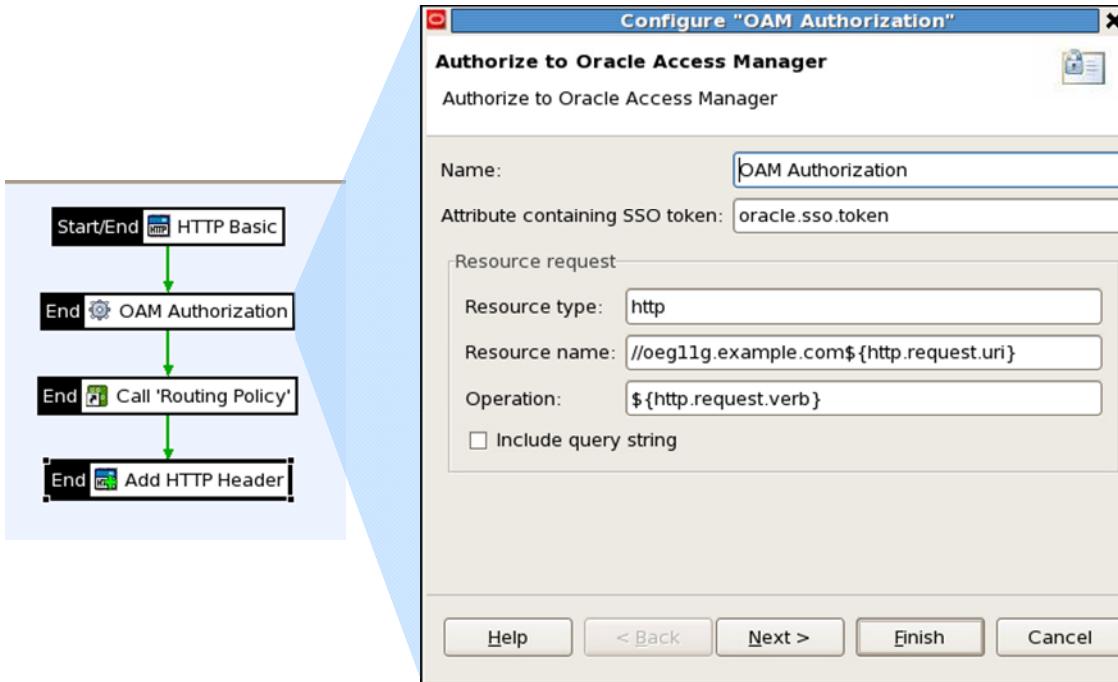
ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The most important settings you need to configure include:

- **Resource type**, which represents the type of resource being requested. The default value is HTTP.
- **Resource name**, which is the name of the resource. Set it to following “//<host-identifier>\${http.request.uri}” where <host-identifier> is the “Host Identifier” value you specified when creating AccessGate entry on the Access Server. The “\${http.request.uri}” means that the value of the path of the incoming HTTP message.
- **Operation**, which is the type of operation to be performed against the resource. When the resource type is HTTP, the possible operations are GET and POST.
- **Single Sign-On** section, where an SSO token can be created and stored in a message attribute, so that it can be added later to the response that the client receives.
- **OAM Access Server SDK directory:** Enter the location where the Access Manager SDK has been installed in the “Oblix installation directory”.

Authentication Policy



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The policy flow is:

- The message will pass through the HTTP Basic filter to authenticate to Oracle Access Manager.
- After the user has been successfully authenticated the message passes through the authorization filter where the user session is authorized for access to the protected web service.
- The message then gets routed to the web service.
- The message response passes through an Add HTTP Header filter to add a valid session token to the header of the message and then gets passed back to the client.

Quiz

Which of the following components must be on the machine, running the OEG so that Gateway can act as a client to OAM:

- a. Oracle Adaptive Access Manager
- b. AccessGate
- c. Access SDK
- d. WebGate



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Roadmap

- Authentication by using OEG and OAM
- Authorization by using OEG and OES



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Fine-grain Versus Coarse-Grain Authorization

- Coarse-grained authorization is less restrictive:
 - Examples:
 - Employees can log in to the application.
 - Non-traders cannot do anything with accounts.
- Fine-grained authorization is more restrictive:
 - Example:
 - AccountTraders can trade accounts if the account firm ID matches a firm ID for the user, the account ID matches the accounts that this user owns, and the request is made during business hours.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Coarse-grain authorization is based on categorizing users based on simple rules and at a relatively high-level. These types of rules do not take many factors into consideration when determining a user's access.

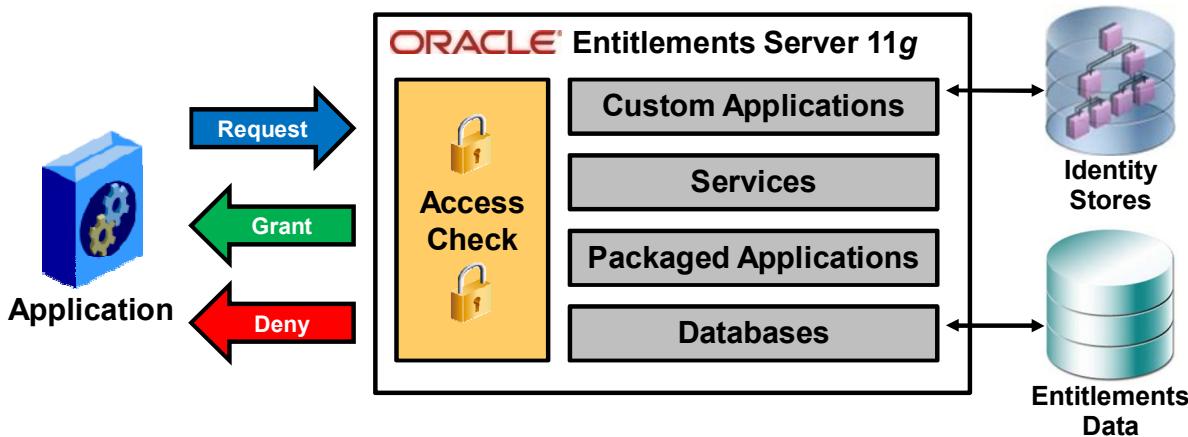
Fine-grain authorization is based on more detailed and specific information that more clearly defines a user and the state of the environment the user is operating in. Often a fine-grain authorization policy first uses coarse-grain methods to narrow down the use case to a certain subset of users (by group or by role), and then uses other attributes to make finer-grain determinations of the user's rights. These attributes can be anything that can be compared or analyzed at run time, such as:

- User attributes that are designed for security purposes
- Application data attributes that can be compared to user attributes
- External system data attributes
- Environmental requirement attributes, such as time of day, day of week, and so on

Using OES for Authorization

OES provides:

- Fine-grained authorization by using roles and attributes
- Centralized policy management
- Distributed run-time enforcement for applications, middleware, databases, and SOA



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE

Oracle Entitlements Server 11g is a product that provides fine-grained authorization services that enable an organization to define and manage policies that control access to application resources. A policy defines access privileges that specify who can do what to which resource, when and how it can be done. Types of resources that can be protected include:

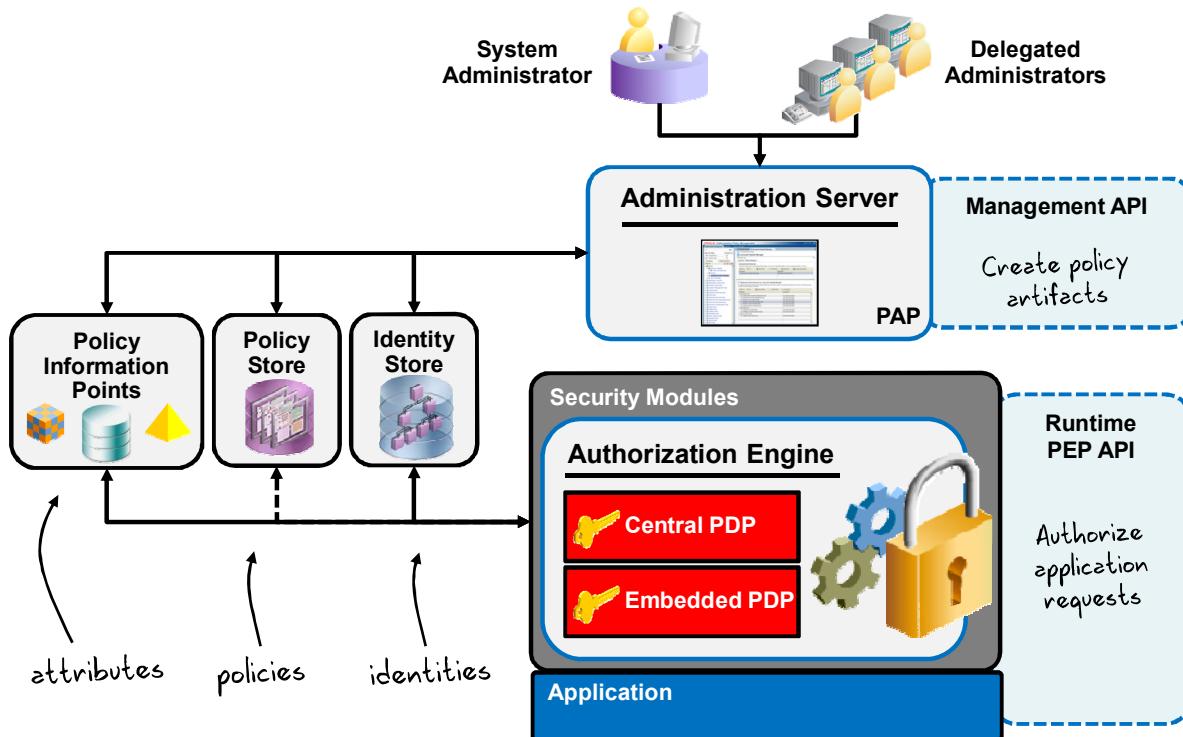
- Software components (URLs, Java Server Pages, Enterprise JavaBeans, methods, servlets and the like used to construct an application)
- Business objects (representations of user accounts, personal profiles and contracts such as trading accounts in a trading application, patient records in a health-care application, or anything used to define a business relationship).

Oracle Entitlements Server 11g:

- Provides centralized policy management
- Supports distributed access control enforcement for all types of resources including software components and application business objects
- Supports a policy model that is easy to administer yet allows for complex sets of conditions under which access can be granted (or denied)
- Operates with LDAP based Identity stores for user and group information.

Note: Oracle Entitlements Server 11g does not provide authentication services.

OES Architecture



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Here is a diagram of the high-level architecture of OES:

Administration Server:

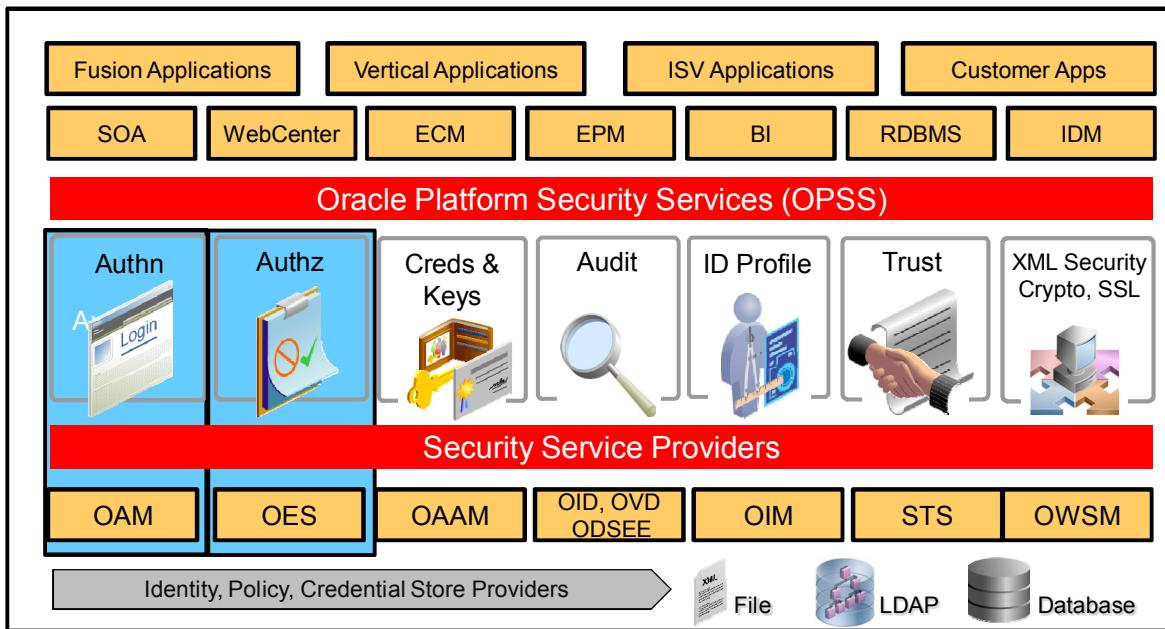
- The Policy Administration Point (PAP) is the OES Administration Server that manages policies and the artifacts related to security. It is connected to other systems that are needed to work with policies:
 - Identity Store:** Separately administered user store
 - Policy Store:** Persistent storage of policies managed by OES
 - Policy Information Points (PIP):** Systems that can provide external attribute data used in OES policies
- The PAP is accessible programmatically using the OES Management API (MAPI).

Security Modules (SM) / Authorization Engine:

- An SM instance comprises the libraries and processes that are commonly referred to as an OES client.
- Depending on the type of SM instance, the authorization engine that is serving as the Policy Decision Point (PDP) is available in either of the following ways:
 - Centrally via a networking port by an SM proxy
 - Locally within the same JVM process, which is referred to as embedded

- Depending on the deployment model of the SM, it may or may not have direct access to the policy store:
 - **Non-Controlled and Controlled-Pull:** The SM has direct access to the policy store, does not cache a copy of the policies locally, and keeps the policy model in memory. Policies are distributed directly either automatically by intervals or programmatically.
 - **Controlled-Push:** The SM does not have a direct connection to the policy store. Policies are distributed manually from the PAP to the SM, which caches a copy of the policies locally on the file system in order to run if the OES Admin Server is not available, and keeps the policy model in memory.
- The authorization engine PDP is accessed by the application by using the Policy Enforcement Point (PEP) API. This API is used to ask the PDP if the user is authorized to take the requested action on a particular resource.
- The Application serves as the PEP, and uses the PEP API to call the PDP for an authorization decision. The PDP returns the outcome to the Application, which then enforces the decision made by the PDP.

OAM, OES and OPSS



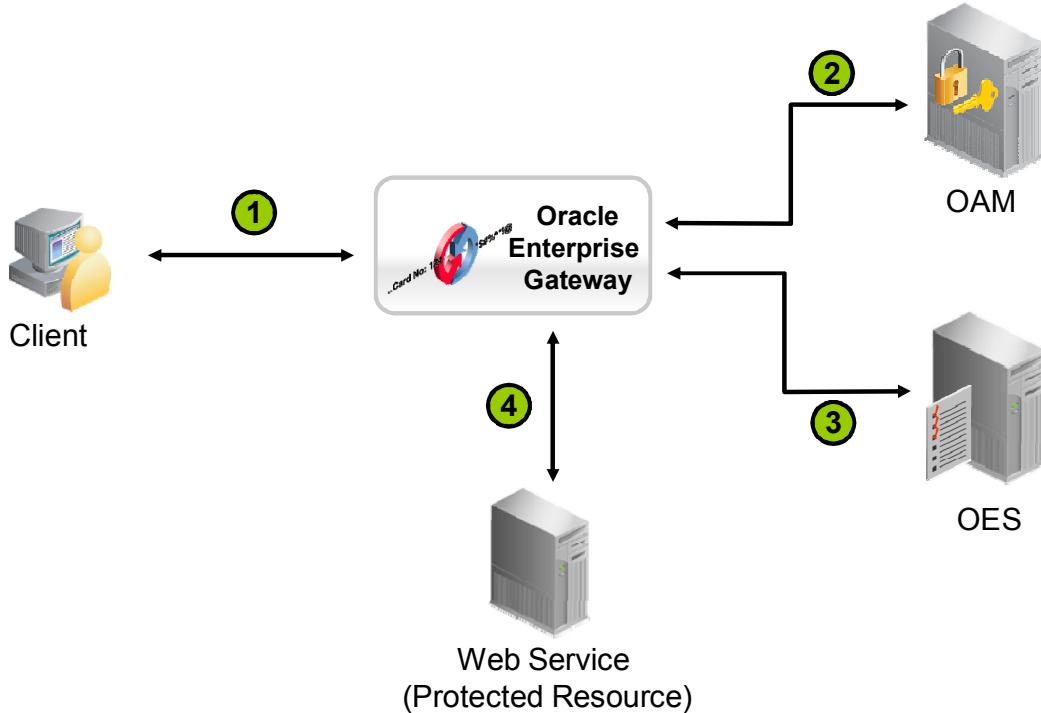
OAM, OES provide the authentication and authorization service functionality.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The diagram illustrates OAM and OES 11g in the context of the Oracle Platform Security Services (OPSS) stack. OPSS provides basic implementations for various services such as authentication, authorization, credential and key services, and auditing, among others. In particular, OAM 11g implements the OPSS Authentication service, and OES 11g implements the OPSS Authorization service.

Authorization By Using OEG and OES



ORACLE®

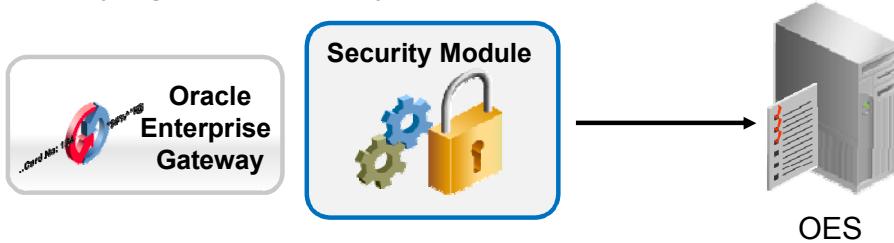
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The above diagram shows the sequence of events that occurs when a client sends a message to that Gateway that needs to be authorized to Oracle Entitlements Server:

1. A client application sends a message containing credentials to the Gateway.
2. The Gateway extracts the credentials and delegates authentication to a third party system (Oracle Access Manager, LDAP, database, CA SiteMinder, RSA Access Manager, and so on).
3. When the client has been authenticated, the Gateway queries Oracle Entitlements Server to see if the specific client is permitted to access the resource (Web Service) that they are trying to contact.
4. When authentication and authorization has passed, the message is trusted and forwarded to the target Web Service.

Connecting to Oracle Entitlements Server

- The OES client installation enables you to:
 - Create the Security Modules (SM) instance
 - Provide the API classpath for OEG
- Perform the following steps to connect the Gateway to OES:
 - Installing OES Client software
 - Configuring the OES Client
 - Modifying the Gateway classpath



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

OES Client is needed for the application's run time to enforce the authorization of policies defined in the server. The OES client installer enables you to create the Security Modules (SM) instances that communicate with the OES server to obtain policy definitions for enforcement and to provide the API classpath for applications which are using OES policies to protect the resources.

To connect the Gateway to Oracle Entitlements Server, you need to perform the three tasks listed in the slide:

- **Installing OES Client:** The OES Client (Security Module) must be installed on the same machine running the Gateway. The OES Client is a separate installation, not included in the Oracle Identity and Access Management 11g Release 1 (11.1.1.5.0).
- **Configuring the OES Client:** Policies are distributed to individual Security Modules that protect applications and services. Policy data is distributed in a controlled manner or in a non-controlled manner. The distribution mode is defined in the `jps-config.xml` configuration file for each Security Module. The specified distribution mode applies to all Application Policy objects bound to that Security Module. Follow the instructions in *Oracle Entitlements Server 11g Integration Guide* to make the configuration. OES Client is needed for the applications runtime to enforce the authorization of policies defined in the OES server.

The Gateway's classpath must be extended to include the OES client JARs on its classpath. To achieve this, create a jvm.xml file at the following location:
`<GATEWAY_DIR>/conf/jvm.xml`

Defining Authorization Policy by Using OES Admin Server

1. Create a new Application.
2. Create a new Resource type.
3. Create a new Resource.
4. Create a new Authorization.
5. Define a new Security Module (SM) definition in OES11g.
6. Bind the Application to the SM.
7. Distribute policy changes.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You need to define an authorization policy in OES Administration Server, and then invoke authorization decisions from OEG. The slide lists the high-level steps to create the authorization policy. For details, refer to Oracle Entitlements Server 11g documentation.

Authorization Policy in OES Admin Server

The screenshot shows the Oracle Entitlements Server Authorization Management console. The left sidebar has a search bar and navigation links for Global, Applications (HelloOESWorld selected), and a central workspace. The central workspace displays the 'Home' page for the HelloOESWorld application. It includes sections for Application Name (HelloOESWorld), Resource Types, Role Mapping Policies, Application Roles, Entitlements, Resources, and Authorization Policies. A note at the bottom says 'Note: objects will be created in the default domain'. At the bottom left is the 'Entitlements Resource Center' with links to get started, navigate, and configure. At the bottom right is the 'Search and Create' section with links to search for External Roles, Applications, and Create Application. The top right corner shows user information: Signed in as weblogic.

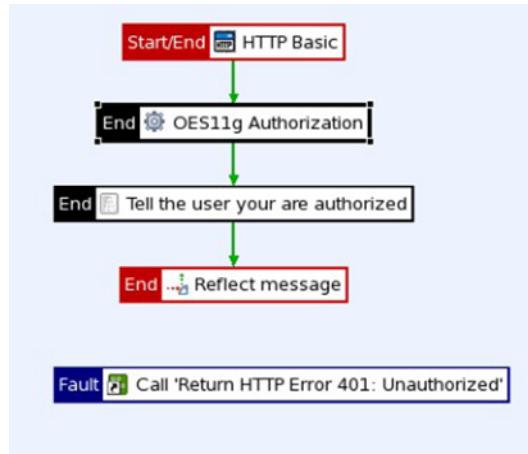
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The slide shows an example of authorization policy created for HelloOESWorld. The authorization policy which gives user `weblogic` privilege to perform `write` on `MyResource`.

Creating Authorization Policy in OEG

- Configure the Authentication filter so that the `authentication.subject.id` attribute is populated with the subject to be used in OES authorization.
- Configure the OES Authorization filter.



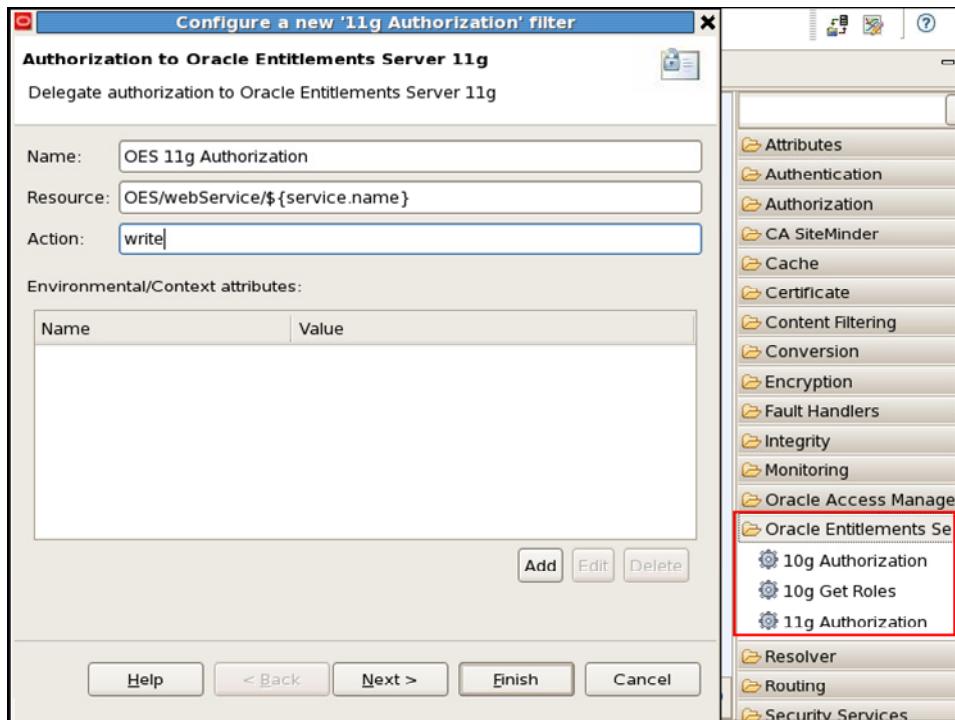
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

After you have the right OES11g configuration, you need to configure the Gateway so that it delegates authorization decisions to the OES. The final policy is shown in the slide. To create the policy, the following steps are required:

- Configure the Authentication filter so that the `authentication.subject.id` attribute is populated with the subject to be used in OES authorization.
- Configure the Oracle Entitlements Server Authorization filter.

Configuring Oracle Entitlements Server Authorization Filter



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The following properties you need to configure for the authorization filter:

- For Resource, enter the resource that is to be authorized. For more information about the format of this value, see [Formatting the Resource String](#) in the OES client documentation. An example of a resource string would be:
OEG/webService/\${service.name}
- For Action, enter the HTTP verb (POST, GET, DELETE, and so on), or if this policy is reused for multiple services, enter the verb as a message attribute that is expanded at run time (for example \${http.request.verb}), or some text that represents the action (write).
- You can optionally configure some environment context attributes.

Quiz

The Security Module instance needs to be created for the applications runtime to enforce the authorization of policies defined in the OES server.

- a. True
- b. False



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe how OEG works with Oracle Access Manager (OAM) for authentication
- Describe how OEG leverages Oracle Entitlements Server (OES) for fine-grained authorization



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Demo Overview: Integrating OEG with OAM and OES

This demo covers the following topics:

- Configuring OEG to Delegate Authentication to OAM 11g
- Configuring OEG to Delegate Authorization to OES 11g



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

15

Securing Services in the Cloud

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe Cloud security risks
- Describe how to secure API keys by using OEG



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Cloud Computing Definition

Cloud computing model:

- Uses Web 2.0
- Is composed of three service models:
 - Software as a Service (SaaS)
 - Platform as a Service (PaaS)
 - Infrastructure as a Service (IaaS)
- Supports four deployment models:
 - Public
 - Private
 - Community
 - Hybrid



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Cloud computing has evolved as a net centric, service-oriented computing model. Consumers purchase computing resources as on-demand basis and get worry free with the underlying technologies used. Cloud computing model is composed of three service models:

- **Software as a Service (SaaS):** Salesforce
- **Platform as a Service (PaaS):** Google Apps
- **Infrastructure as a Service (IaaS):** Amazon EC2

Cloud computing supports four deployment models: Public, Private, Community, and Hybrid. A third-party service provider, stores and maintains data, application or infrastructure of Cloud user. Relinquishing the control over data and application poses challenges of security, performance, availability, and privacy.

Security Risks of Cloud Computing

- Use of the public Internet
- External data storage
- Multi-tenancy



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Use of the public Internet, External data storage, Multi-tenancy, On-demand, and Elastic are the characteristics of Cloud computing, but some of them may pose security risks.

- **Use of the public Internet:** Unclear and unaddressed accountability; loss and misuse of data
- **Multi-tenancy:** Insufficient logging and monitoring; inadequate separation of data between different customers
- **External data storage:** Weak control of data; legal complications

Application Programming Interface (API) and API Keys

- APIs are interfaces to services in the Cloud.
- API keys control the access to APIs.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

APIs are the rules that determine how applications interface with cloud-side service offerings to enable enterprises to reach far beyond their own web properties to distribute data, content or services that have relevance to their business operations. Effectively these APIs are the interface to the business services and access to APIs is controlled by API keys. API keys are codes generated to control and manage access to these services and most organizations use some form of API keys to access their cloud services.

API keys control access to business sensitive information, also can be called cloud assets (for example, email, sales leads, or shared documents) and pay-as-you-use Cloud services. As such, if an organization casually manages API keys they are at risk of:

- Unauthorized individuals using the keys to access confidential information
- The possibility of huge bills for unapproved access to pay-as-you-use Cloud services

Security Issues Faced by Cloud Computing

- Supplying an audit trail to individual users
- Protecting the Application Programming Interface (API) keys

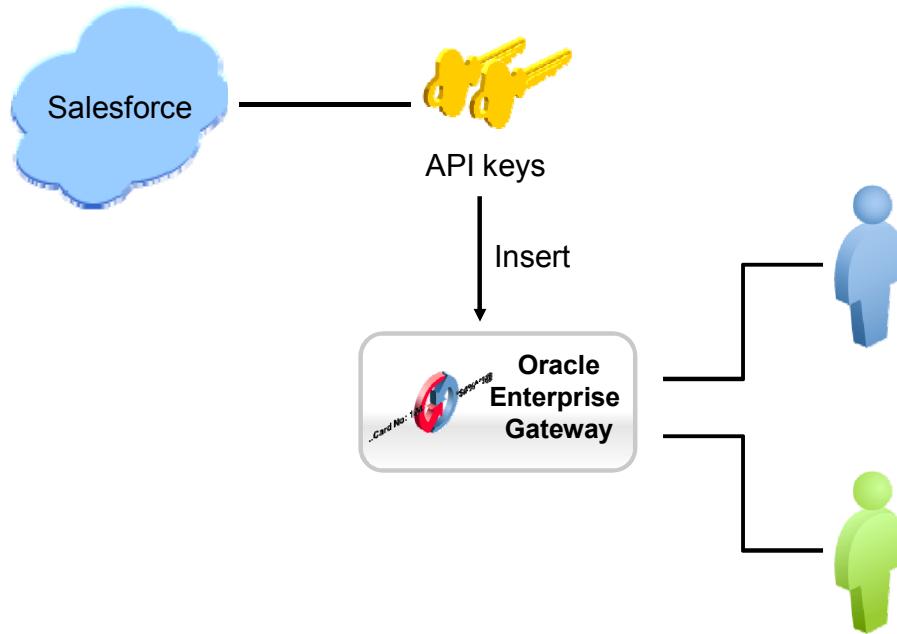


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

There are some security issues IT managers are facing when they want to use cloud computing:

- In the multi-tenant cloud, one problem is how to supply an audit trail to individual customers, who are probably running their workloads on a server with many fellow cloud users.
- Like SSL keys, API keys are codes generated to control and manage access to services in cloud. API keys would offer a direct access to your cloud, hence the data stored in it. Therefore, protecting the API keys has become crucial as well.

Centrally Protecting and Managing API Keys By Using OEG



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Cloud-based services use API key for identification and signing. Any application looking to integrate with the Cloud service will need access to the API key. Distributing API keys from the various Cloud services to all the on-premise applications is operationally difficult and insecure. It is best to store the API keys centrally at the Gateway and delegate key management and signing to the Gateway. Storing keys by using the Gateway's on-board or network based HSM options further improves security.

Summary

In this lesson, you should have learned how to:

- Describe Cloud security challenges
- Describe how to secure API keys by using OEG



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.