Oracle® Retail Service Backbone

Security Guide

Release 16.0.027

E94819-01

May 2018

Oracle Retail Service Backbone Security Guide, Release 16.0.027

E94819-01

Primary Author: Sanal Parameswaran

Contributing Author: Nathan Young

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

Send Us Your Comments

Oracle Retail Service Backbone Security Guide, Release 16.0.027

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at `http://www.oracle.com`.

Preface

Designing and implementing security in Web services presents an interesting set of challenges, owing to the fact that they are distributed and loosely coupled in nature. This is true with Retail Service Backbone (RSB), which is built on SOA and OSB infrastructure. The RSB system comprises of Web services that are loosely coupled, distributed environments that allow retailers to integrate heterogeneous applications within the enterprise or expose business functions to their customers and partners over the internet.

Implementing security at an enterprise level depends heavily on key security requirements of an enterprise. Your involvement is essential in making the right decisions. This security guide supplements other documentation to ensure that the information is complete, coherent, and detailed.

This document serves as security guideline to the implementation personnel and developers. The content in this document is structured in a way to give a high level view of the security landscape in general and focuses on Oracle Retail certified security strategies and the toolsets with detailed instructions to implement them.

Audience

The target audience of this document are system architects, developers, security implementation personnel, integrators, and trainers. The content of the document requires a good understanding of the WebLogic application server, Web services security concepts, and a high level understanding of RSB system.

Documentation Accessibility

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

Access to Oracle Support

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

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 16.0) or a later patch release (for example, 16.0.027). If you are installing the base release and additional patch releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail Documentation on the Oracle Technology Network

Oracle Retail product documentation is available on the following Web site:

http://www.oracle.com/technology/documentation/oracle_retail.html

(Data Model documents are not available through Oracle Technology Network. You can obtain them through My Oracle Support.)

Conventions

The following text conventions are used in this document:

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

1

General Security Principles

Security is about risk management and implementing effective countermeasures. One of the most important concepts in security is that effective security is a combination of people, process, and technology.

This chapter discusses a list of Web service security terms that are part of a coherent framework defined in the Web services architecture. The relationships between the terms are defined in the concepts and relationships section of the definitions in this document.

Web Services Security or WS-Security

It is for developers who wish to expose their own services in an environment that requires protection of messages from being tampered or read in transit, or in situations in which the sender must be positively identified. The term WS-Security is usually used to refer to a group of specifications that handle encryption and digital signatures, enabling you to create a secure application.

Aspects of Web Services Security

This section covers the various aspects of Web services security.

Authentication

The authentication process verifies that you are who you claim to be. Your identity is verified based on the credentials that you present, such as username/password, digital certificate, standard Security Assertion Markup Language (SAML) token, or Kerberos token. In the case of Web services, credentials are presented by a client application on your behalf.

Authorization (or Access Control)

Granting access to specific resources based on an authenticated user's entitlements or specific role (e.g., corporate buyer).

Confidentiality (or Privacy)

Confidentiality is keeping information secret. Personally Identifiable Information (PII) or confidential business data could be present in a Web service request or response messages. Confidentiality of such data can be achieved by encrypting the content of the request or response messages using the XML Encryption standard.

Integrity (or Non Repudiation)

Integrity is making sure 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 timestamp ensuring that a transaction can't be later repudiated by either the sender or the receiver. XML messages are signed using the XML signature standard.

SOA Security in Practice

Effective SOA security in practice includes the following measures:

- Coordinating people, process, and technology.
- Integrating and leveraging various levels of standards (general security standards, XML security standards, Web services security standards).
- Integrating and leveraging various user stores and role stores.
- Making trade-offs between user experience, technical, and business perspectives.

WS-Security Standards / Web Services Security Concepts

This section covers the various aspects of Web services security standards.

Standard Industry Approach

There are numerous ways to build or implement a secured service to protect the SOA infrastructure against attack. Standards allow policies to be applied to SOA, thus ensuring controlled usage and monitoring and provide security ramifications in enterprise integration. Standards such as WS-Security, SAML, WS-Trust, WS-Secure Conversation, and WS-SecurityPolicy focus on the security and identity management aspects of SOA implementations that use Web services.

The WS-* architecture is a set of standards-based protocols designed to secure Web service communication. For more information on these WS-* security standards, refer to the following sections.

WS-Policy

WS-Policy allows Web services to define policy requirements for endpoints. These requirements include privacy rules, encryption rules, and security tokens.

WS-Security

WS-Security allows Web services to apply security to Simple Object Access Protocol (SOAP) messages through encryption and checks integrity on all or part of the message.

WS-Trust

WS-Trust allows Web services to use security tokens to establish trust in a broken security environment.

WS-SecureConversation

WS-SecureConversation builds on top of WS-Policy, WS-Security, and WS-Trust to enable secure communications between a client and a service.

WS-ReliableMessaging

WS-ReliableMessaging allows Web services and clients to trust that when a message is sent, it will be delivered to the intended party.

WS-AtomicTransactions

WS-AtomicTransactions allows transaction-based Web services in which transactions can be rolled back in the event of a failure.

Oracle WSM Principles

Oracle WSM is a standards-based solution that allows you to externalize Web services security and management from the applications. Oracle WSM provides declarative security and management through pre-defined policies.

Oracle WSM is based on three main operations: Define, Enforce, and Monitor.

Define

Define consists of attaching security and management policies to the Web services to be protected. Examples of policies are Authenticate Request messages using username/password, Decrypt Messages using WS-Security, and Sign Response messages.

Enforce

Enforce is the ability provided by Oracle WSM to distribute policies from a central Policy Manager to several Policy Enforcement Points (PEP) or Agents that locally execute security and management policies at runtime.

Monitor

Monitor is the tracking (in graphical charts) of the runtime security and management events captured by the Oracle WSM enforcement points.

Types of Security

To secure a WebLogic Web service, configure one or more of three different types of security.

Table 1–1 Web Services Security

Security Type	Description
Transport-level security	SSL is used to secure the connection between a client application and the Web service.
Message-level security	Data in a SOAP message is digitally signed or encrypted. It may also include identity tokens for authentication.
Access control security	Specifies which roles are allowed to access Web services.

Security Options in WebLogic

This section discusses the security options available in WebLogic.

Transport-level Security

Transport-level security secures the connection between the client application and the WebLogic server with Secure Sockets Layer (SSL). SSL provides secure connections by allowing two applications, connecting over a network, to authenticate the other's identity and by encrypting the data exchanged between the applications. Authentication allows a server, and optionally a client, to verify the identity of the application on the other end of a network connection. A client certificate (two-way SSL) can be used to authenticate you. Encryption makes data transmitted over the network intelligible only to the intended recipient. Transport-level security includes HTTP BASIC authentication as well as SSL.

Message-level Security

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 is digitally signed and encrypted, rather than just the connection. And finally, you can specify that only individual parts or elements of the message be signed, encrypted, or required.

Transport-level security, however, secures only the connection itself. This means that if there is an intermediary between the client and the WebLogic 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. 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 SOAP message be encrypted. Message-level security can also include identity tokens for authentication.

Access Control Security

Access control security answers the question “who can do what?”. At first, you specify the security roles that are allowed to access a Web service. A security role is a privilege granted to users or groups based on specific conditions. Then, when a client application attempts to invoke a Web service operation, the client authenticates itself to the WebLogic server, and if the client has the authorization, it is allowed to continue with the invocation. Access control security secures only the WebLogic server resources. That is, if you configure only access control security, the connection between the client application and WebLogic server is not secure and the SOAP message is in plain text.

Note: In future releases of Oracle Service Bus (OSB), Oracle Web Services Manager policies will replace and enhance the functionality of WLS 9 security policies. While this version of Oracle Service Bus (12c or 12.2.1.3.0) continues to support WLS 9 policies, you should consider using Oracle Web Services Manager policies for new service creation to prepare for the eventual deprecation of WLS 9 policy support.

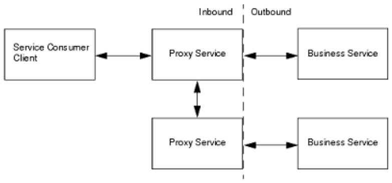
It is recommended that you use Oracle WSM policies whenever possible. You cannot mix your use of Oracle WSM and WebLogic Web service policies.

You can attach only one type of security policy to a Web service, either WebLogic server security policies or Oracle WSM policies. You cannot attach both WebLogic server policies and Oracle WSM policies to the same Web service, through either the annotation mechanism, the Administration Console, Fusion Middleware Control, or a combination of the three.

You can attach an Oracle WSM WS-Security policy only to a JAX-WS Web service. You cannot attach this type of policy to a JAX-RPC Web service.

If you are establishing security requirements for a new business service that uses Web services security, Oracle recommends that you require clients to provide SAML tokens. SAML is the emerging standard for propagating user identities within Web services.

Oracle Services Bus Security



Inbound Security

Inbound security ensures that Oracle Service Bus proxy services handle only the requests that come from authorized clients. By default, any anonymous or authenticated user can connect to a proxy service. It can also ensure that no unauthorized user has viewed or modified the data sent from the client. Proxy services can have two types of clients: service consumers and other proxy services.

Outbound Security

Outbound security secures communication between a proxy service and a business service. Most of the tasks that you complete for outbound security is for configuring proxy services to comply with the transport-level or message-level security requirements that business services specify. For example, if a business service requires username and password tokens, you create a service account, which either directly contains the username and password, passes along the username and password that was contained in the inbound request, or provides a username and password that depends on the username that was contained in the inbound request.

Options for Identity Propagation

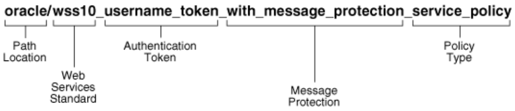
A key decision that must be made when designing security for Oracle Service Bus is how to handle (propagate) the identities that clients provide. You can configure Oracle Service Bus to do any of the following:

- Authenticate the credentials that clients provide
- Perform authorization checks
- Pass client credentials to business services unchanged
- Map client credentials to a different set of credentials that a business service can authenticate and authorize
- Bridge between security technologies

Security Policies

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

Identifying the Different Parts of a Policy Name



When Should You Use Oracle WS-Security Policies?

Consider the following scenarios:

If you develop WebLogic server JAX-WS Web services or clients that interact with SOA Composite Services, ADF Components, or WebCenter Services, then you should use the Oracle WSM WS-Security policies.

If you develop only WebLogic server native Java JAX-WS Web services, then you should use WebLogic server's WS-Security policies.

Are WebLogic Policies and Oracle WSM Policies compatible?

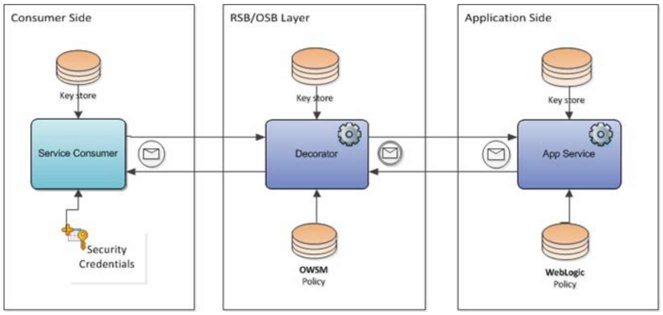
A subset of WebLogic Web service policies inter-operate with Oracle WSM policies. That is, for specific policy instances, you can attach an Oracle WSM policy to the Web service client or service, and an Oracle WebLogic server policy to the WebLogic Java EE Web service or client, and they will inter-operate.

2

Retail Service Backbone Security

This chapter provides an overview of security in the Retail Service Backbone (RSB) and discusses how security applies to it.

The RSB is an enterprise infrastructure for service oriented communication. RSB provides an architectural approach which involves Oracle Retail applications being exposed as “Service Providers”, and OSB core components called “Decorator Services” wrap the edge-app-services to provide virtualized and operational visibility to Oracle Retail services infrastructure. Lastly, the decorator services expose proxy WSDL's that are called by “Service Consumers” which could be Retail applications or third party applications capable of making a SOAP invocation.



The RSB key components can be loosely coupled across platform, technology, and physical topologies. In order to establish an end-to-end security in RSB system, one has to secure all the intermediary layers that are part of a communication channel through RSB. As discussed earlier, there are three main layers, Service Provider, Decorator and the Service Consumer. Each of these must participate in configuring security in order to achieve effective security while making Web service calls.

Security decisions are mostly Service Provider driven; Decorators and Consumers will need to adhere to the chosen security mechanism by the provider in an integrated system. RSB provides numerous tools to ease the security configuration and automate the process which otherwise is quite cumbersome.

Recommended Security Approach for RSB

As RSB embeds networks of interconnected Web services, it does present key challenges, particularly in terms of security and administration. Oracle Retail, at an enterprise level, will not try to dictate customers on the security policy they must use in their enterprise. Oracle Retail will have security recommendations for our applications.

RSB services security principles must be able to fit into the customer's predefined corporate security policies and guidelines. Oracle Retail applications test, certify, and document a few commonly used security policies that work out-of-the-box with minimal effort.

You are free to reconfigure to a higher or lower level of Web service security than that certified by Oracle Retail. In such situations, Oracle Retail does not provide setup/configuration documentation. You should follow the Fusion Middleware documentation and set up the system on your own. The security configuration is supported (as long as Fusion Middleware supports it) but not certified by Oracle Retail.

Securing an RSB system involves following components. Each of these components is a layer in the RSB security architecture.

1. Edge App Services
2. Decorator Business Services
3. Decorator Proxy Services

Edge App Services

These services are deployed as part of ear files on the WebLogic server and it is not required to have Fusion Middleware components in that environment. In such environments, OWSM is not available to secure Web services. Therefore, these services are secured using the WebLogic Web service policies.

Decorator Services

The decorator proxy and business services are deployed in an OSB server where OWSM components can be used for Web service policies. Therefore, decorator proxy and business services use those OWSM policies which are compatible with the WebLogic policies applied to edge app services.

The policy files for business and proxy services depend on which policy has been attached to the corresponding edge app service. Business service needs to be configured with an OWSM client policy file which is corresponding to the policy file attached to the edge app service. The proxy services need to be configured with an OWSM service policy file that should have configuration matching with edge app policy file. The client of the proxy service needs to use a client policy file which matches the service policy for OSB proxy service.

Table 2–1 Decorator Services

Proxy Service Client	Proxy Service	Business Service	App Service
OWSM client policy	OWSM service policy	WebLogic compatible policy	WebLogic Web service policy

Active-Intermediary

The proxy service processes the header in the client's SOAP messages and enforces the message-level access control policy on the messages.

For example, a client encrypts and signs its SOAP message and sends it to a proxy service. The proxy service decrypts the message and verifies the digital signature, then routes the message. Before the proxy service sends the response back to the client, the proxy service signs and encrypts the message. The client then decrypts the message and verifies the proxy service's digital signature.

Pass-Through

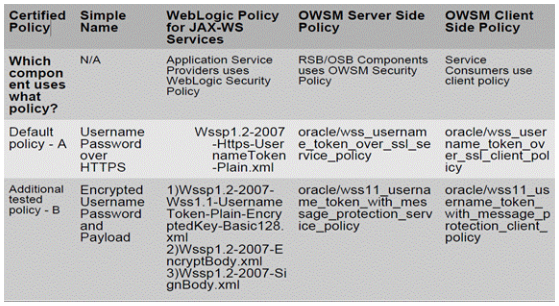
Instead of processing the header in the client's SOAP messages, the proxy service passes the message untouched to a business service. Although the proxy service does not process the secured sections of the SOAP message, it can route the message based on values in the header. When the business service receives the message, it processes the security header and acts on the request. Note that the business service must use the Web Services Policy (WS-Policy) framework to describe which of its operations are secured with message-level security. The business service sends its response to the proxy service, and the proxy service passes the response untouched to the client.

For example, the client encrypts and signs the message and sends it to the proxy service. The proxy service does not decrypt the message or verify the digital signature; it simply routes the message to the business service. The business service decrypts the messages and verifies the digital signature, and then processes the request. The response path is similar.

Oracle Retail Enterprise Level Out-of-the-Box Certified Policies

There are two types of security configurations certified for RSB by Oracle Retail. For ease of use these are classified and referred in this document as Policy A and Policy B.

Each of these certified approaches involve a set of security policies. The following table describes the required set of policies for Policy A and B.



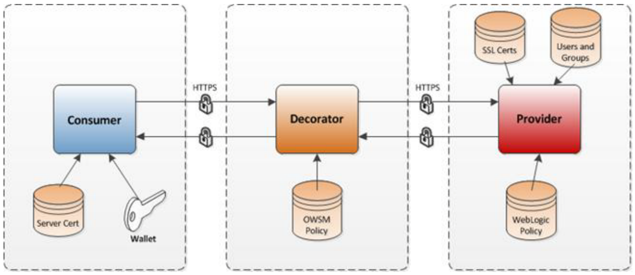
Certified Policy	Simple Name	WebLogic Policy for JAX-WS Services	OWSM Server Side Policy	OWSM Client Side Policy
Which component uses what policy?	N/A	Application Service Providers uses WebLogic Security Policy	RSB/OSB Components uses OWSM Security Policy	Service Consumers use client policy
Default policy - A	Username Password over HTTPS	Wssp1.2-2007-Https-UsernameToken-Plain.xml	oracle/wss_username_token_over_ssl_service_policy	oracle/wss_username_token_over_ssl_client_policy
Additional tested policy - B	Encrypted Username Password and Payload	1)Wssp1.2-2007-Wss1.1-UsernameToken-Plain-EncryptedKey-Basic128.xml 2)Wssp1.2-2007-EncryptBody.xml 3)Wssp1.2-2007-SignBody.xml	oracle/wss11_username_token_with_message_protection_service_policy	oracle/wss11_username_token_with_message_protection_client_policy

Service providers are the endpoint services which are invoked by the decorator proxy and business services; therefore edge app services must be secured first and based on the message-level and transport-level authentication required by them, the proxy and business services are secured with similar configurations.

Policy A (Default Policy)

Policy A is a Username Token over HTTPS. This security configuration is referred to as Policy A in this document.

- Wssp1.2-2007-Https-UsernameToken-Plain.xml



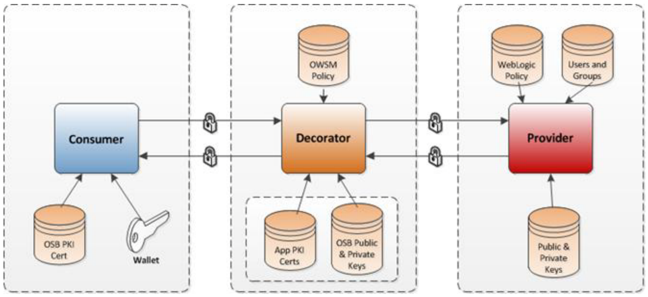
Policy A is selected as the default policy for the following reasons:

- Most widely (SSL) used policy
- Easy to configure and setup
- Data going over the wire is encrypted and is secure
- Interoperable between WebLogic and OWSM policy
- Good performance for the level of security provided
- Good support from all programming languages

Policy B

Policy B is a Message Protection plus username token. This security configuration is referred to as Policy B in this document.

- Wssp1.2-2007-Wss1.1-UsernameToken-Plain-EncryptedKey-Basic128
- Wssp1.2-2007-EncryptBody
- Wssp1.2-2007-SignBody



It is highly recommended that the personnel who implement security carefully consider their options and take a strategic decision that benefits their business.

The following facts are true for the encrypted username/password/payload Policy B:

- Payload is encrypted/decrypted in the SOAP infrastructure layer and not in the TCP/IP layer.
- The policy is complicated to set up in the WebLogic server and OWSM.
- Since encryption and decryption is happening higher up in the stack, the performance is slower.
- Limited support from other programming languages.

3

Installation Overview

This chapter provides an overview of the installation framework for the RSB.

Installing Infrastructure Components

RSB provides the framework and toolset for centralized software product lifecycle management of RSB components. All configuration and management tasks are done from a single centralized location defined as rsb-home using specific tools that support all phases of the RSB product lifecycle. The framework and the toolset are collectively referred to as RSB Kernel or RSB Builder tool.

For information on RSB Kernel and installation details, see the *Oracle Retail Service Backbone Installation Guide* and the *Oracle Retail Service Backbone Implementation Guide*.

Assumptions / Prerequisites

RSB security principles are based on the following assumptions:

- Participating Retail Applications or external applications are hosted on a WebLogic server.
- The Web services are secured using WebLogic policies (as opposed to OWSM policies).
- If the application services are secured with any policy other than what is mentioned in this document or custom policies, the instructions in the document do not work.
- The security setup in the document does not address authorization. Authorization must be taken care by the individual application hosting the services.
- <DomainHome> is the home directory of the WebLogic domain where the application services are deployed.

Secure Installation and Configuration

Security configurations for Web services are generally performed after the services are deployed. The application services need to be secured before security configurations can be applied in the integration layer (RSB). There are no programmatic changes needed for securing the application services.

Administrative access to the application server is required for completing the security configuration of the application services.

Security configurations must be executed with accuracy. A small mistake in the security configuration setup can make the whole system nonfunctional.

Troubleshooting of security related issues are generally harder than troubleshooting of application issues. Considering this, Oracle recommends that you verify the configuration after every step. Oracle provides “sample” scripts and detailed instructions to minimize the chances of error.

Application teams may need to modify the sample scripts to suit their needs and/or incorporate similar concepts in their install scripts/process.

Security configuration comprises broadly of three phases, each must be performed sequentially as described. The following sections describe each phase and the detailed steps that you need to take for a successful configuration. Most of the steps have been automated and can be achieved by running the scripts provided. However, there are some required manual steps.

> **Note:** Due to known vulnerabilities, Oracle recommends disabling SSLv3 in all products. We recommend using the TLSv1.2 protocol. WebLogic server can be configured to use the TLSv1.2 protocol by adding the following line in the setDomainEnv.sh. Restart the server after making the change.
>
> ```
> JAVA_OPTIONS=”$JAVA_OPTIONS
> -DwebLogic.security.SSL.minimumProtocolVersion=TLSv1
> .2”
> ```

Pre-Installation Steps

Perform the following steps regardless of the policy (A or B) you are using to secure the services. Once these are done, perform the policy specific tasks described in the following sections.

1. Download and extract the RTG provided **RsbAppServiceSecuritySetupSamplesPak16.0.027ForAll16.0.027Apps.zip**. The zip file contains the sample scripts to help you configure the application server for Web service security. The structure of the extracted zip file should be as shown in the figure below:

```
security-setup-home/
     service-provider/
         config/
            security-setup/
                 app-lib
                 lib
     service-consumer/
```

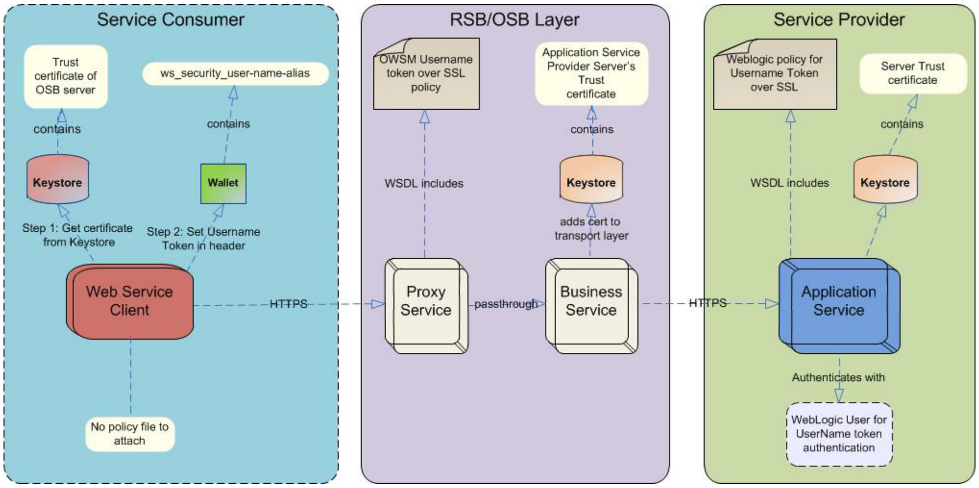
2. Copy the contents of **security-setup-home/service-provider** directory to **<DomainHome>/config** directory of the WebLogic application server where your RSE generated Web service is deployed.

```
scp security-setup-home/service-provider/* <user>@<your host>:/<your
path>/<DomainHome>/config/
```

3. Run **setDomainEnv.sh** in the current shell. (In order to run a script in the current shell, enter a dot and space before the script. This is required to make the variables set in the script be available to the current shell.)

```
.  <DomainHome>/bin/setDomainEnv.sh
```

Configuring Security for Policy A


Service Consumer
Trust certificate of OSB server
ws_security_user-name-alias
contains
contains
Keystore
Wallet
Step 1: Get certificate from Keystore
Step 2: Set Username Token in header
Web Service Client
HTTPS
No policy file to attach
RSB/OSB Layer
OWSM Username token over SSL policy
Application Service Provider Server's Trust certificate
contains
WSDL includes
Keystore
adds cert to transport layer
Proxy Service
passthrough
Business Service
HTTPS
Service Provider
Weblogic policy for Username Token over SSL
Server Trust certificate
contains
WSDL includes
Keystore
Application Service
Authenticates with
WebLogic User for UserName token authentication

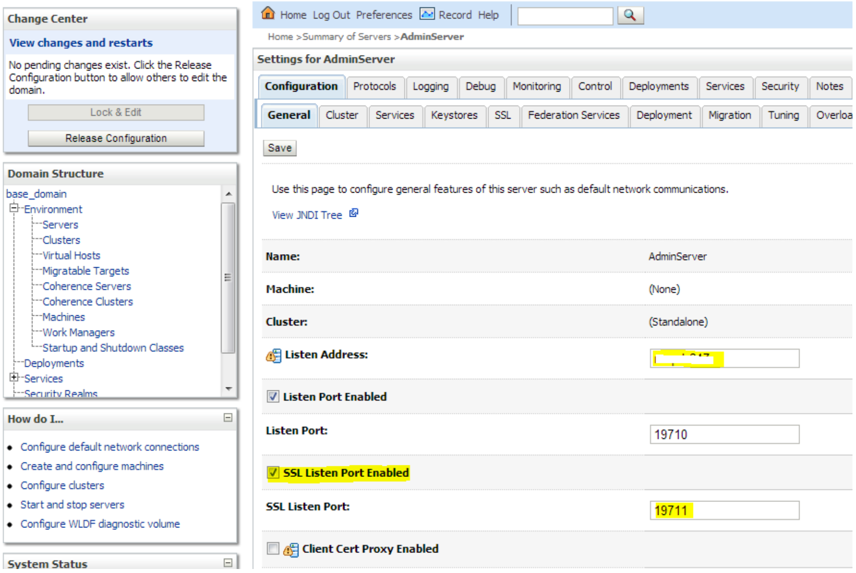
Securing Edge Application Services with Policy A

This section describes the steps you need to take to secure the edge application services with Policy A.

Enabling the HTTPS Port

To enable the HTTPS port, take the following steps:

1. In WebLogic Admin Console, click Environment --> Servers.
2. Click the server where the Web service is deployed.
3. Click the Configuration -- > General tab.
4. Check the **SSL Listen Port Enabled** check box. Do not uncheck the **Listen Port Enabled** check box. The OSB needs both the HTTP and SSL listen ports enabled.
5. Enter a port number for the SSL Listen Port. This is the port number for the service endpoint.
6. Enter the hostname in **Listen Address** field.
7. Click **Save**.



No pending changes exist. Click the Release Configuration button to allow others to edit the domain.

Use this page to configure general features of this server such as default network communications.

Name:	AdminServer
Machine:	(None)
Cluster:	(Standalone)
Listen Address:	[redacted]
☑ Listen Port Enabled	
Listen Port:	19710
☑ SSL Listen Port Enabled	
SSL Listen Port:	19711
☐ Client Cert Proxy Enabled	

Configuring and Using Authentication

If your credentials are stored in an external repository (e.g., LDAP), follow the WebLogic documentation to make these credentials available to the WebLogic Web service infrastructure so that it can authenticate you during Web service invocation. If you are testing with users in the built-in user registry of WebLogic, you may follow the instructions below.

You can add a user using the provided script or through the WebLogic Admin Console.

- If you are using the script, run the following command from the security-setup-home created during pre-installation steps:

  ```
  cd security-setup-home/service-provider
  app-service-security-config.sh -add-user
  ```

- The script prompts for the following:
  - username
  - password
  - WebLogic admin URL
  - WebLogic Admin username
  - WebLogic Admin password

The username and password from the service consumer is authenticated against the username and password entered in this step. For manual steps to create users in the WebLogic server, refer to the WebLogic 12.2.1.3.0 documentation.

Configuring and Using Security Policies

This section describes steps to configure and use security policies.

Note: To secure an EJB app, it must be deployed with custom role and policies. During application deployment, after selecting the target server, select the "Custom Roles and Policies" option under Security settings. Security policies (Policy A and Policy B) can be attached to edge app Web services using either the provided script or by attaching the policies manually.

Configuring Security Policies for Policy A Setup - Script Method

Take the following steps:

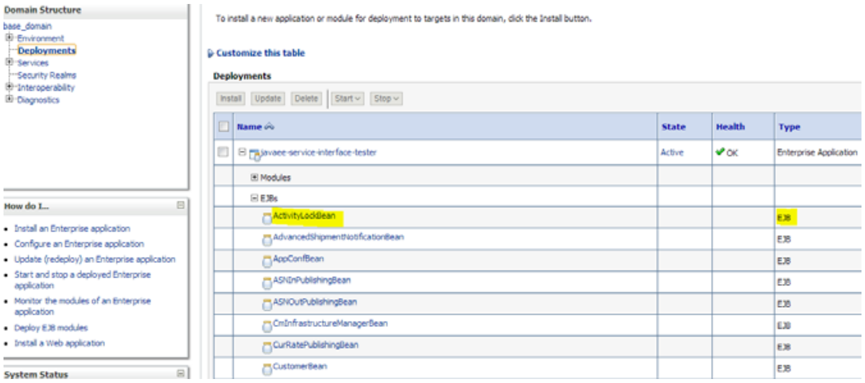
1. Copy the ear files of the Web service provider EJBs to the **`service-provider/config/security-setup/app-lib`** folder.
2. Run **`app-service-security-policy-condition-config.sh -u <WebLogic Admin Server URL> -a -e <Enterprise Application Name> -t user -n <user>`**.
3. Change directory using **`cd security-setup-home/service-provider`**.
4. Run **`app-service-security-policy-config.sh -u <WebLogic Admin Server URL> -a -e <Enterprise Application Name> -p PolicyA`**.

The usage of the script is displayed when the script is run with the –usage parameter. The script extracts the ear file and uses all the jars from the ear file in classpath. It attaches the Web service policies corresponding to Policy A to the services defined in the ear file.

Configuring Security Policies for Policy A Setup - Manual Method

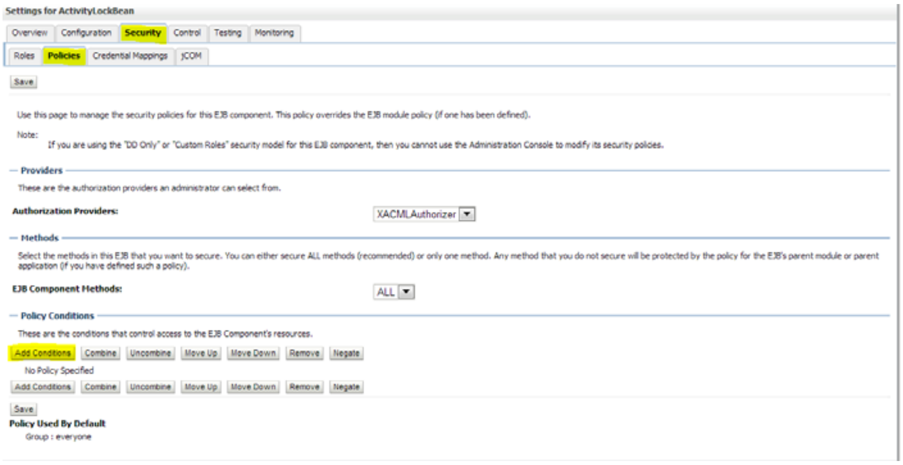
For each of the services to be secured, perform the following steps in WebLogic Admin Console. Make sure you attach the user created in the Configuring and Using Authentication section to the EJB.

1. Click **Deployments**. Click on the EJB (not the Service) you want to secure and then click **Security** --> **Policies**.


Domain Structure
base_domain
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
How do I...
Install an Enterprise application
Configure an Enterprise application
Update (redeploy) an Enterprise application
Start and stop a deployed Enterprise application
Monitor the modules of an Enterprise application
Deploy EJB modules
Install a Web application
System Status
Customize this table
Deployments
Install
Update
Delete
Start
Stop

To install a new application or module for deployment to targets in this domain, click the Install button.

□	Name	State	Health	Type
□	javaee-service-interface-tester	Active	✓ OK	Enterprise Application
	Modules			
	EJBs			
	ActivityLockBean			EJB
	AdvancedShipmentNotificationBean			EJB
	AppConfBean			EJB
	ASNInPublishingBean			EJB
	ASNOutPublishingBean			EJB
	CmInfrastructureManagerBean			EJB
	CurRatePublishingBean			EJB
	CustomerBean			EJB



Use this page to manage the security policies for this EJB component. This policy overrides the EJB module policy (if one has been defined).

Note:
If you are using the "DD Only" or "Custom Roles" security model for this EJB component, then you cannot use the Administration Console to modify its security policies.

These are the authorization providers an administrator can select from.

Authorization Providers: XACMLAuthorizer

Select the methods in this EJB that you want to secure. You can either secure ALL methods (recommended) or only one method. Any method that you do not secure will be protected by the policy for the EJB's parent module or parent application (if you have defined such a policy).

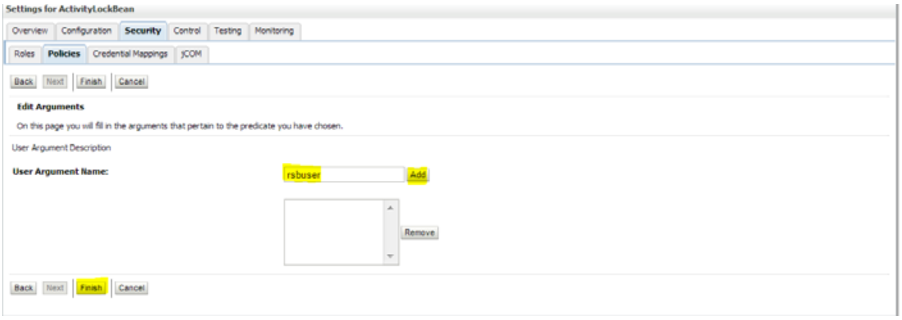
EJB Component Methods: ALL

These are the conditions that control access to the EJB Component's resources.

No Policy Specified

Policy Used By Default
Group : everyone

2. Click **Add Conditions** --> **Predicate List**. Pick User from the dropdown list and then click **Next** --> User Argument Name. Type the username you created and then click **Add** --> **Finish** --> **Save**.

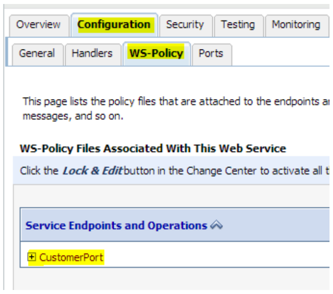


On this page you will fill in the arguments that pertain to the predicate you have chosen.

User Argument Description

User Argument Name: rsbuser

3. Attach policy to the service and navigate to Configuration tab --> WS-Policy --> click on the service port.

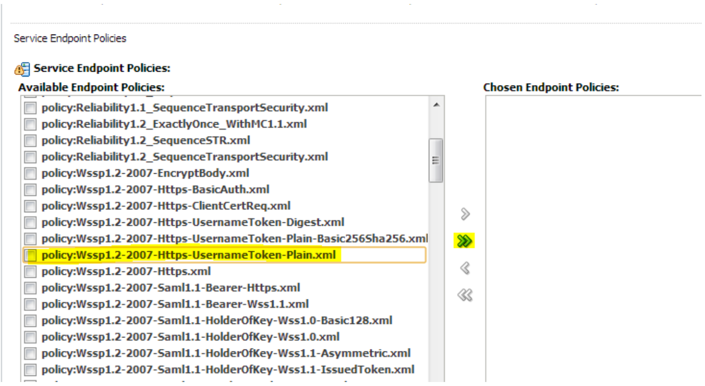


This page lists the policy files that are attached to the endpoints a
messages, and so on.

WS-Policy Files Associated With This Web Service

Click the *Lock & Edit* button in the Change Center to activate all t

4. Pick **WebLogic** --> **Next** --> **Service Endpoint Policies**. Select policy: **`Wssp1.2-2007-Https-UsernameToken-Plain.xml`** and then click **Finish**.



Service Endpoint Policies

Service Endpoint Policies:

Available Endpoint Policies:

- □ policy:Reliability1.1_SequenceTransportSecurity.xml
- □ policy:Reliability1.2_ExactlyOnce_WithMC1.1.xml
- □ policy:Reliability1.2_SequenceSTR.xml
- □ policy:Reliability1.2_SequenceTransportSecurity.xml
- □ policy:Wssp1.2-2007-EncryptBody.xml
- □ policy:Wssp1.2-2007-Https-BasicAuth.xml
- □ policy:Wssp1.2-2007-Https-ClientCertReq.xml
- □ policy:Wssp1.2-2007-Https-UsernameToken-Digest.xml
- □ policy:Wssp1.2-2007-Https-UsernameToken-Plain-Basic256Sha256.xml
- □ policy:Wssp1.2-2007-Https-UsernameToken-Plain.xml
- □ policy:Wssp1.2-2007-Https.xml
- □ policy:Wssp1.2-2007-Saml1.1-Bearer-Https.xml
- □ policy:Wssp1.2-2007-Saml1.1-Bearer-Wss1.1.xml
- □ policy:Wssp1.2-2007-Saml1.1-HolderOfKey-Wss1.0-Basic128.xml
- □ policy:Wssp1.2-2007-Saml1.1-HolderOfKey-Wss1.0.xml
- □ policy:Wssp1.2-2007-Saml1.1-HolderOfKey-Wss1.1-Asymmetric.xml
- □ policy:Wssp1.2-2007-Saml1.1-HolderOfKey-Wss1.1-IssuedToken.xml

Chosen Endpoint Policies:

5. Click **OK** if WebLogic asks to save **`Plan.xml`**.

6. Restart the WebLogic server.

7. Verify the configuration by checking the WSDL of the service. The WSDL must have the policy information in it.

Securing Decorator Services with Policy A

Decorator services can be secured using rsb-home if their edge application service is secured using one of the supported security configurations. If the edge-app security configuration is not one of the supported security configuration, then you need to follow the fusion middleware documentation for security and secure the proxy and business services manually from the OSB console.

Configuring Security Using rsb-home

As described earlier in the Installing Infrastructure Components section, the complete RSB lifecycle management including security configuration and maintenance must be performed using the tool set packaged as rsb-home.

Configuring Security (In Brief) Using rsb-home

- There are no changes to the decorator deployment process. Initially the decorators are deployed as unsecured services. After deploying decorator services, it is recommended that you test the services to make sure that they are able to invoke the edge application services and the flows are working as expected. The https configurations in the **`rsb-deployment-env-info.properties`** must be correctly configured with the https settings. For more information, see the *Oracle Retail Service Backbone Installation Guide*.

- Once the services are verified, secure the edge application services as described in the Securing Edge Application Services with Policy A section.

- After securing the edge application services, secure the business and proxy services. rsb-home consists of scripts to configure security for the decorator services. When these scripts are executed, the following events take place in the background:

  a. The edge application service WSDL downloads from the URL which is available in **`rsb-deployment-env-info.properties`** file.

  b. The WSDLs contain information about the policy files that are used to secure the Web service.

  c. Based on the policy, the script determines which OWSM client policy is suitable for business service and which OWSM service policy is suitable for a proxy service. This can be achieved only for the security configurations that are supported by RSB. If the policy file of edge application service does not match with the supported configurations, then the script throws an error that the decorator services cannot be secured using the script and they should be secured manually following the fusion middleware security documentation.

d. If the security configuration is one of the supported configurations, then the proxy and business service files in the decorator jar are updated with the changes that are required for that security configuration.

The structure of the directory is shown below. The scripts responsible for security configuration are shown in **bold**/***bold-italics*** and the folders that stage the security related files are shown in *italics*.

```
rsb-home
|
|---- deployment-home
| |---- bin
| |     |---- configure-rsb-app-server-for-security-policy-b.sh
| |     ---- rsb-deployer.sh
| |---- conf
| |     |---- ddl
| |     | `---- RSB_INFRASTRUCTURE_SCHEMA_DEFINITION.SQL
| |     |---- rsb-decorator-instrumentation.properties
| |     |---- rsb-decorator-service-to-family-name-association.properties
| |     |---- rsb-deployment-env-info.properties
| |     `---- rsb-integration-flows.xml
| `---- log
|---- download-home
| |---- all-app-service-decorator
| |---- all-business-process-service-decorator
| |---- all-functional-business-process
| |---- all-functional-service-int-flow
| |---- bin
| |     `---- check-version-and-unpack.sh
| |---- integration-guide
| `---- log
|---- integration-lib
`---- service-assembly-home
    |---- app-service-decorator
    |---- bin
    |     |---- download-app-service-wsdl.sh
    |     |---- generate-rsb-decorator-security-config.sh
    |     |---- rsb-compiler.sh
    |     `---- setup-message-protection-security-credentials.sh
    |---- business-process-orchestration
    |---- business-process-service-decorator
    |---- conf
            |---- default-rib-files
            |  |---- rib-func-artifacts
            |  |---- rib-global
            |  |---- rsb-admin-internal
            `----
app-service-policy-id-to-decorator-service-policy-id-map.properties
    |     |---- logging.properties
    |     `---- rsb-builder-internal-trust-store.jks
    |---- integration-guide
    |---- log
    |---- service-integration-flow
    `---- service-policy-config
            |---- input
```

```
| |---- app-service-provider-wsdl
| `---- security-policy
`---- output
       |---- app-service-provider-security-policy
       |---- decorator-service-biz-security-policy
       |---- decorator-service-consumer-client-security-policy
       |---- decorator-service-proxy-security-policy
       | `---- service-name-to-policy-id-map.properties
       `---- decorator-service-proxy-wsdl
```

For more information on securing the decorator services, see the *Oracle Retail Service Backbone Installation Guide*.

Consumer Side Configuration for Policy A

For invoking services over SSL, consumer applications must be able to validate service provider server certificate. For more information on installing certificates on a WebLogic server, see the WebLogic documentation.

After the certificates are installed, the consumer applications should have access to a wallet file. This wallet file contains the username and password required for user authentication by the Web services.

Refer to Appendix A for a sample Java code for invoking a service over SSL. In the sample code, the username and password have been hard-coded for simplicity. But in real applications, it is recommended that you use a wallet file.

Note: For access to the wallet file, you may use the CSM package that Oracle provides.

Consumer Side Configuration for Policy A (PLSQL Apps)

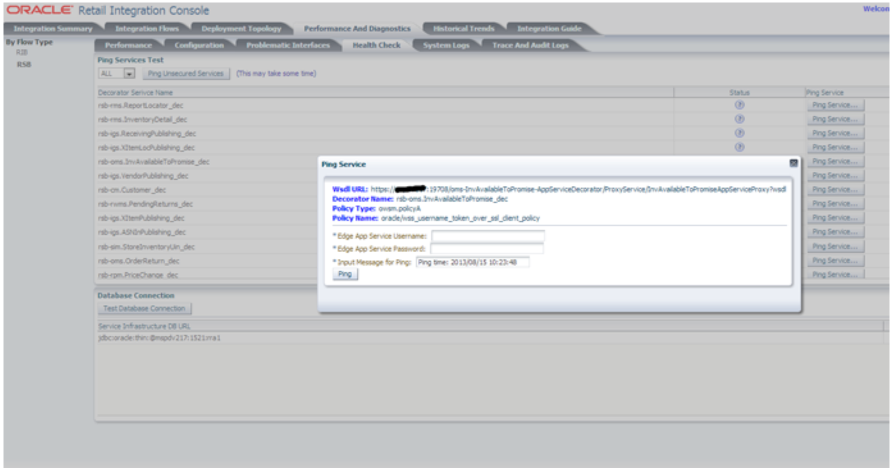
If the service consumer is a PLSQL based application as opposed to a Java application, the security is configured from inside the database. The PLSQL consumer invokes the service. The Retail SOA Enabler (RSE) tool accesses the database and generates PLSQL consumer sample code. It describes step-by-step process to invoke a secured service from a PLSQL client in PLSQLServiceConsumer_ReadMe packaged inside the RSE generated consumer artifacts. For more information, see the *Oracle Retail Integration Bus Service-Oriented Architecture Enabler Tool Guide*.

Post Installation Steps

After the successful configuration of security, verify it before running the real functional tests. Verification can be done by invoking a ping operation of a RSB decorator service securely in the following ways:

1. Using the Retail Integration Console (RIC)
2. Using a SOAP User Interface

Verify Policy Using the Retail Integration Console



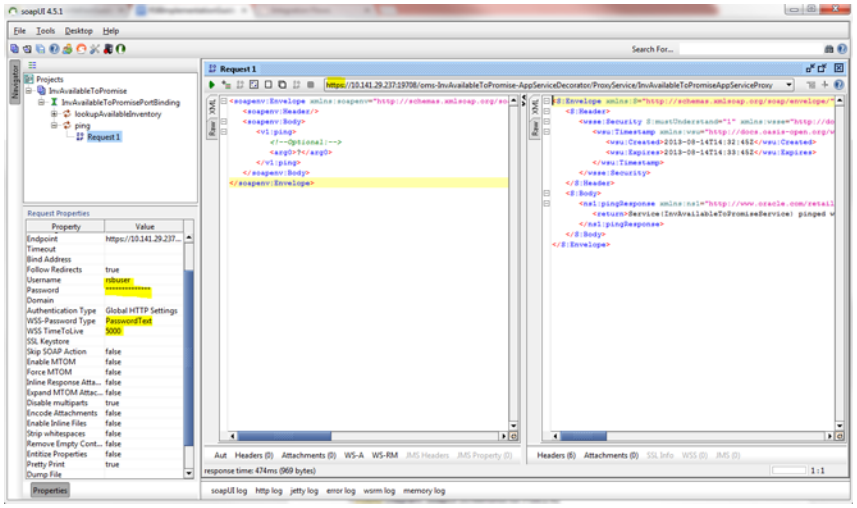
Decorator Service Name	Status	Ping Service
rsb-rms.ReportLocator_dec	ⓟ	Ping Service...
rsb-rms.InventoryDetail_dec	ⓟ	Ping Service...
rsb-igs.ReceivingPublishing_dec	ⓟ	Ping Service...
rsb-igs.XItemLocPublishing_dec	ⓟ	Ping Service...
rsb-oms.InvAvailableToPromise_dec		Ping Service...
rsb-igs.VendorPublishing_dec		Ping Service...
rsb-cm.Customer_dec		Ping Service...
rsb-rwms.PendingReturns_dec		Ping Service...
rsb-igs.XItemPublishing_dec		Ping Service...
rsb-igs.ASNInPublishing_dec		Ping Service...
rsb-sim.StoreInventoryUin_dec		Ping Service...
rsb-oms.OrderReturn_dec		Ping Service...
rsb-rpm.PriceChange_dec		Ping Service...

Wsdl URL: https://[redacted]:19708/oms-InvAvailableToPromise-AppServiceDecorator/ProxyService/InvAvailableToPromiseAppServiceProxy?wsdl
Decorator Name: rsb-oms.InvAvailableToPromise_dec
Policy Type: owsm.policyA
Policy Name: oracle/wss_username_token_over_ssl_client_policy

* Edge App Service Username:
* Edge App Service Password:
* Input Message for Ping: Ping time: 2013/08/15 10:23:48

Service Infrastructure DB URL
jdbc:oracle:thin:@mspdv217:1521:ma1

Verify Policy Using a SOAP User Interface



https://10.141.29.237:19708/oms-InvAvailableToPromise-AppServiceDecorator/ProxyService/InvAvailableToPromiseAppServiceProxy

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/so
   <soapenv:Header/>
   <soapenv:Body>
      <v1:ping>
         <!--Optional:-->
         <arg0>?</arg0>
      </v1:ping>
   </soapenv:Body>
</soapenv:Envelope>
```

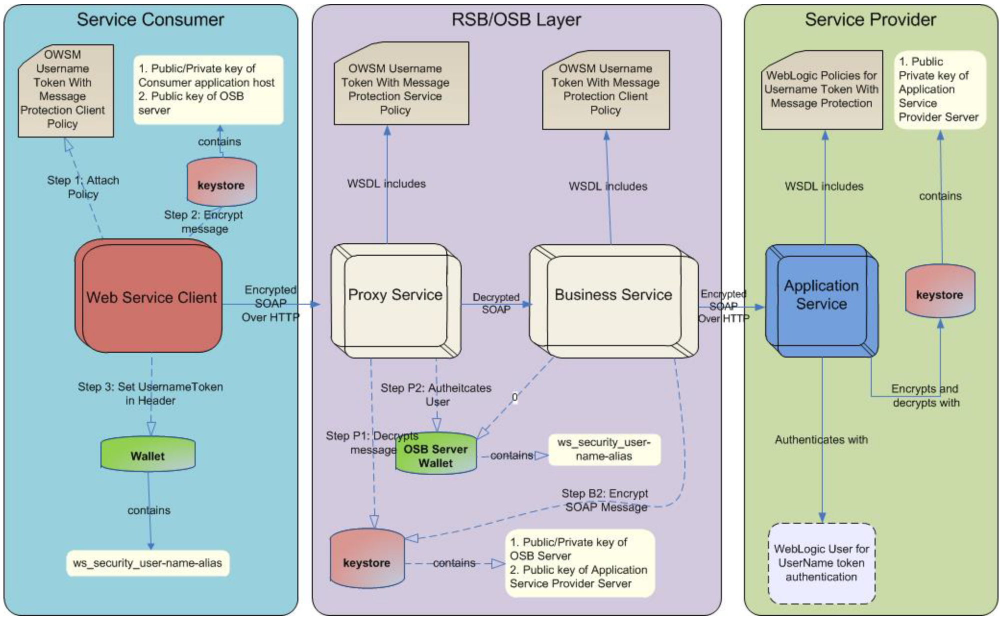
```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
   <S:Header>
      <wsse:Security S:mustUnderstand="1" xmlns:wsse="http://do
         <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/w
            <wsu:Created>2013-08-14T14:32:45Z</wsu:Created>
            <wsu:Expires>2013-08-14T14:33:45Z</wsu:Expires>
         </wsu:Timestamp>
      </wsse:Security>
   </S:Header>
   <S:Body>
      <ns1:pingResponse xmlns:ns1="http://www.oracle.com/retail
         <return>Service(InvAvailableToPromiseService) pinged w
      </ns1:pingResponse>
   </S:Body>
</S:Envelope>
```

Request Properties

Property	Value
Endpoint	https://10.141.29.237...
Timeout	
Bind Address	
Follow Redirects	true
Username	rsbuser
Password	***************
Domain	
Authentication Type	Global HTTP Settings
WSS-Password Type	PasswordText
WSS TimeToLive	5000
SSL Keystore	
Skip SOAP Action	false
Enable MTOM	false
Force MTOM	false
Inline Response Atta...	false
Expand MTOM Attac...	false
Disable multiparts	true
Encode Attachments	false
Enable Inline Files	false
Strip whitespaces	false
Remove Empty Cont...	false
Entitize Properties	false
Pretty Print	true
Dump File	

response time: 474ms (969 bytes)

Configuring Security for Policy B (Message Protection)


Service Consumer
OWSM Username Token With Message Protection Client Policy
1. Public/Private key of Consumer application host 2. Public key of OSB server
contains
Step 1: Attach Policy
keystore
Step 2: Encrypt message
Web Service Client
Encrypted SOAP Over HTTP
Step 3: Set UsernameToken in Header
Wallet
contains
ws_security_user-name-alias
RSB/OSB Layer
OWSM Username Token With Message Protection Service Policy
OWSM Username Token With Message Protection Client Policy
WSDL includes
WSDL includes
Proxy Service
Decrypted SOAP
Business Service
Encrypted SOAP Over HTTP
Step P2: Autheitcates User
0
Step P1: Decrypts message
OSB Server Wallet
contains
ws_security_user-name-alias
Step B2: Encrypt SOAP Message
keystore
contains
1. Public/Private key of OSB Server 2. Public key of Application Service Provider Server
Service Provider
WebLogic Policies for Username Token With Message Protection
1. Public Private key of Application Service Provider Server
WSDL includes
contains
Application Service
keystore
Encrypts and decrypts with
Authenticates with
WebLogic User for UserName token authentication

Securing Edge Application Services with Policy B

This section describes the steps that you can take to secure an edge application service using Policy B.

Generating Key Store and Certificates

Generate public and private keys and key store for message encryption and digital signing.

1. Download RsbAppServiceSecuritySetupSamplesPak<version>ForAll<version>Apps_eng_ga.zip
2. Unzip and copy the scripts from /security-setup-home/service-provider to the <Edge-app-domain-home>/config folder.
3. Navigate to <Edge-app-domain-home>/bin folder and execute. SetDomainEnv.sh

> **Note:** WebLogic server should be up and running.

4. Run the script located in <Edge-app-domain-home>/config to generate necessary certificates, keys, and key store.

```
app-service-security-config.sh -generate-cert
```

Enter the key store password and private key password.

Note down these passwords for use in the next step.

> **Note:** You should obtain certificates from CA in production environments. Generated certificates should be used only for development and demo purposes.

Configuring the WebLogic Server to Use the Certificates

1. Configure the WebLogic server to use the certificate, keys, and key store to protect messages.

   Note: WebLogic server should be up and running.

   `app-service-security-config.sh -config-wls-cert-keystore`

   Enter the required inputs:

   a. WebLogic admin URL, for example: t3://<hostname>:<port>

   b. Username, password

   c. Key store password

   d. Private key password

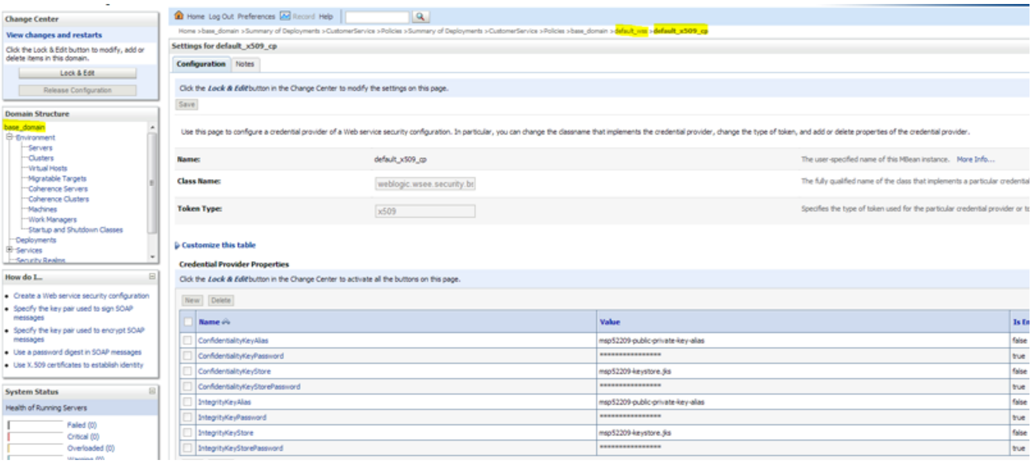
   Note: Keystore and private key password should be same as provided in previous step.

   ---

2. If needed, restart the WebLogic server after the configuration updates.

3. Verify that above configuration reflects in WebLogic console.

   In the WebLogic Admin console, click the domain name in Domain Structure. Then click Web Service Security --> default_wss --> default_x509_cp. The configuration screen is displayed.


Change Center
View changes and restarts
Lock & Edit
Release Configuration
Domain Structure
base_domain
Environment
Servers
Clusters
Virtual Hosts
Migratable Targets
Coherence Servers
Coherence Clusters
Machines
Work Managers
Startup and Shutdown Classes
Deployments
Services
Security Realms
How do I...
Create a Web service security configuration
Specify the key pair used to sign SOAP messages
Specify the key pair used to encrypt SOAP messages
Use a password digest in SOAP messages
Use X.509 certificates to establish identity
System Status
Health of Running Servers
Failed (0)
Critical (0)
Overloaded (0)
Home Log Out Preferences Record Help
Home >base_domain >Summary of Deployments >CustomerService >Policies >Summary of Deployments >CustomerService >Policies >base_domain >default_wss >default_x509_cp
Configuration
Notes
Save
Customize this table
New
Delete
More Info...

Click the Lock & Edit button to modify, add or delete items in this domain.

Settings for default_x509_cp

Click the *Lock & Edit* button in the Change Center to modify the settings on this page.

Use this page to configure a credential provider of a Web service security configuration. In particular, you can change the classname that implements the credential provider, change the type of token, and add or delete properties of the credential provider.

Name: default_x509_cp — The user-specified name of this MBean instance.

Class Name: weblogic.wsee.security.bs — The fully qualified name of the class that implements a particular credential

Token Type: x509 — Specifies the type of token used for the particular credential provider or to

Credential Provider Properties

Click the *Lock & Edit* button in the Change Center to activate all the buttons on this page.

	Name	Value	Is En
□	ConfidentialityKeyAlias	msp52209-public-private-key-alias	false
□	ConfidentialityKeyPassword	****************	true
□	ConfidentialityKeyStore	msp52209-keystore.jks	false
□	ConfidentialityKeyStorePassword	****************	true
□	IntegrityKeyAlias	msp52209-public-private-key-alias	false
□	IntegrityKeyPassword	****************	true
□	IntegrityKeyStore	msp52209-keystore.jks	false
□	IntegrityKeyStorePassword	****************	true

Configuring and Using Authentication

If your credentials are stored in an external repository (e.g., LDAP), follow the WebLogic documentation to make these credentials available to WebLogic Web service infrastructure so that it can authenticate you during the Web service invocation. If you are testing with users in the built-in user registry of WebLogic, follow the instructions below.

You can add a user using the provided script or through the WebLogic Admin console. If you are using the script, run the following command:

Navigate to <Egde-app-domain-home>/config folder, and execute

`app-service-security-config.sh -add-user`

The script prompts you for the following:

- Username and Password (for user authentication)
- WebLogic Admin URL, for example: t3:// <host name>:<port>
- WebLogic Admin username and password

The username and password from the service consumer is authenticated against the username and password entered in this step.

Configuring and Using Security Policies

Security policies (Policy A and Policy B) can be attached to edge app Web services using either the provided script or by attaching the policies manually.

Configuring and Using Security Policies - Script Method

Take the following steps:

1. Copy the application ear file (e.g.: javaee-service-interface-tester-<version>.ear) of the Web service providers EJBs to <Egde-app-domain-home >/config/security-setup/app-lib
2. Change directory using below command cd security-setup-home/service-provider/config.
3. Execute app-service-security-policy-condition-config.sh -u <WebLogic Admin Server URL> -a -e <Service Name> -t user -n <user>.

   For example: sh app-service-security-policy-condition-config.sh -u t3:// <host name>:<port> -a -e customerservice -t user -n rsbuser

   > **Note:** The above example will apply policy conditions to only selected web services. If user wants to apply policy conditions to all the web services, execute the below step.

   sh app-service-security-policy-condition-config.sh -u <WebLogic Admin Server URL> -a -e <Enterprise Application Name> -t user -n <user>
4. Run app-service-security-policy-config.sh -u <WebLogic Admin Server URL> -a -e <Enterprise Application Name> -p PolicyB.

   For example: sh app-service-security-policy-config.sh -u t3:// <host name>:<port> -a -e javaee-service-interface-tester -p PolicyB
5. Restart admin and manage server.

The usage of the script is displayed when the script is run with the –usage parameter. The script extracts the ear file and uses all the jars from the ear file in classpath. It attaches the Web service policies corresponding to Policy B to the services defined in the ear file.

> **Note:**
>
> - Verify that user is added in ejb and policies are configured in webservice or open WSDL to see policies embedded in that.
> - If not, configure policies manually in Weblogic.

Configuring Security Policies for Policy B Setup - Manual Method ;

To secure each of the services, perform the following steps in WebLogic Admin Console:

1. Attach the user, created in the Configuring and Using Authentication section, to the service.

   Click **Deployments** and then click on the EJB you want to secure. Click **Security** --> **Policies** --> **Add Conditions** --> Predicate List. Pick the user from the dropdown list and click **Next** --> **User Argument Name**. Type the username you created and then click **Add** --> **Finish** --> **Save**.

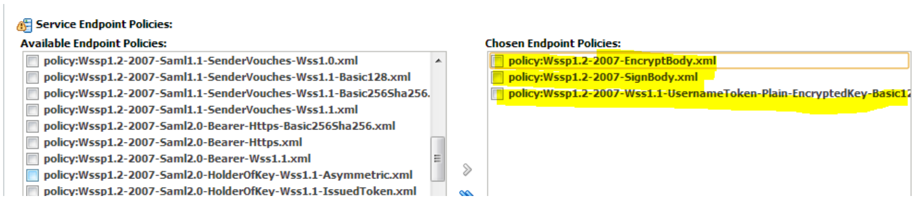
2. Attach the three policies to the service.

   Navigate to the **Configuration** tab --> **WS-Policy**. Click on the service port and pick WebLogic -->**Next** --> In Service Endpoint Policies. Select the following policies from available policy list:

   policy:Wssp1.2-2007-Wss1.1-UsernameToken-Plain-EncryptedKey-Basic128.xml

   policy:Wssp1.2-2007-EncryptBody.xml

   policy:Wssp1.2-2007-SignBody.xml



Service Endpoint Policies:

Available Endpoint Policies:

☐ policy:Wssp1.2-2007-Saml1.1-SenderVouches-Wss1.0.xml
☐ policy:Wssp1.2-2007-Saml1.1-SenderVouches-Wss1.1-Basic128.xml
☐ policy:Wssp1.2-2007-Saml1.1-SenderVouches-Wss1.1-Basic256Sha256.
☐ policy:Wssp1.2-2007-Saml1.1-SenderVouches-Wss1.1.xml
☐ policy:Wssp1.2-2007-Saml2.0-Bearer-Https-Basic256Sha256.xml
☐ policy:Wssp1.2-2007-Saml2.0-Bearer-Https.xml
☐ policy:Wssp1.2-2007-Saml2.0-Bearer-Wss1.1.xml
☐ policy:Wssp1.2-2007-Saml2.0-HolderOfKey-Wss1.1-Asymmetric.xml
☐ policy:Wssp1.2-2007-Saml2.0-HolderOfKey-Wss1.1-IssuedToken.xml

Chosen Endpoint Policies:

☐ policy:Wssp1.2-2007-EncryptBody.xml
☐ policy:Wssp1.2-2007-SignBody.xml
☐ policy:Wssp1.2-2007-Wss1.1-UsernameToken-Plain-EncryptedKey-Basic12

3. Click **Finish**.

4. Restart WebLogic servers.

5. Verify the configuration by checking the WSDL of the service. The WSDL must have the information of all the three policies in it.

Exporting the Certificate

Export the certificate from the application server. Copy the certificate file to the RSB WebLogic server.

Use the following command to export the certificate:

```
app-service-security-config.sh -export-cert
```

You will have to provide the key store password. The exported certificate file name can be <hostname>-certificate.der.

rsb-home and OSB Side

The following provide an easy reference to the important steps that you need to take related to Policy B that should be run from rsb-home.

For more information on steps to secure the decorator services with Policy B, see the *Oracle Retail Service Backbone Installation Guide*.

Before Policy B, it is assumed that RSB Domain is configured.

> **Note:** Follow rsb-installation guide for RSB Configuration.

1. Make sure that the endpoint URL is correct in the RSB deployment properties file and run the script to download app service WSDL.

   ```
   cd
   <rsb-home>/service-assembly-home/bin/download-app-service-wsdl.sh
   ```

2. Run the following script to populate rsb-home/service-assembly-home/service-policy-config/output/decorator-service-proxy-security-policy/ service-name-to-policy-id-map.properties file.

   ```
   cd <rsb-home>/service-assembly-home/bin
   ```

   Run **generate-rsb-decorator-security-config.sh**

3. Copy the PKI generation utility shell script from rsb-home to the OSB service server machine's <domain_home>/config folder.

   ```
   cd <rsb-home>/integration-lib/rsb-tools/scripts
   ```

   Run **scp generate-pki-certificate-keystore-for-osb.sh <user>@<host>:<rsb_domain>/config**

4. Login to the OSB service server machine. Change directory to <domain_home>/config and run **generate-pki-certificate-keystore-for-osb.sh**.

   ```
   cd <domain_home>/bin
   ```

   ```
   . setDomainEnv.sh
   ```

   ```
   cd ../config
   ```

   ```
   generate-pki-certificate-keystore-for-osb.sh
   ```

5. Copy <hostname>-certificate.der from AppService server's <domain-home>/config to OSB server's <domain-home>/config folder. Run the following command to import the application service's public key in the OSB key store with alias name <appName>-<AppServiceHostName>-remote-host-public-key-alias.

   ```
   cd <rsb-home>/integration-lib/rsb-tools/scripts
   ```

   ```
   scp
   import-remote-server-public-key-certificate-into-keystore.sh
   <user>@<host>:<domain-home>/config
   ```

   Login to OSB Server.

   ```
   cd <domain-home>/config
   ```

   Copy the certificate from the application server to OSB server.

   ```
   scp
   <user>@<host>:<app-server-domain>/config/<host>-certificate.der
   ```

   ```
   import-remote-server-public-key-certificate-into-keystore.sh
   <appName> <appServiceHostName>
   ```

   For example,

```
import-remote-server-public-key-certificate-into-keystore.sh
igs <rsbhost>
```

6. Restart the OSB Server.

7. Run the following command which asks for the key store password, password for OSB server key alias, password for remote server's key alias, and username/password for usernametoken authentication for the Web services. The passwords for these aliases must match with the passwords that are used in the key store.

   ```
   cd <rsb-home>/service-assembly-home/bin
   ```

   ```
   setup-message-protection-security-credentials.sh
   ```

   Script will prompt for below passwords to store in wallet file.

   - keystore password
   - public private key alias passwordremote host key alias password
   - username and password configured in edge app

   The first alias is the keystore-csf-key where the username is pre-populated with the same name as the alias name. The password for this entry should be the key store password of OSB server.

   The second alias is the <hostname>public-private-key-alias where hostname is the OSB server. The username is pre-populated with the same name as the alias name and password of the private key in the key store.

   The third alias is the <appName>-<AppServiceHostName>-remote-host-public-key-alias. The username is pre-populated with the same name as the alias name and password of the private key in the key store. This step is repeated for all the applications for which services are secured with Policy B.

   The fourth alias is <appName>-user-alias. Here the username and password are prompted. The values must match with the username/password configured in application service's WebLogic server for UsernameToken authentication. This step is repeated for all the applications for which services are secured with Policy B.

8. Run the following command to create the user in the OSB WebLogic server and also update the domain-level wallet file with aliases of OSB and application service keys.

   ```
   cd <rsb-home>/deployment-home/bin
   ```

   ```
   configure-rsb-app-server-for-security-policy-b.sh
   ```

9. Copy **/security-setup-home/service-provider/config/app-service-security- config.sh** from **RsbAppServiceSecuritySetupSamplesPak16.0.027ForAll16.0.027Apps_eng_ga.zip** to the <domain-home>/config directory of the OSB server where the OSB services would be deployed.

   Export the certificate from the OSB application server. Copy the certificate file to the Consumer side WebLogic server. Use the following command to export the certificate:

   ```
   app-service-security-config.sh -export-cert
   ```

   You will have to provide the key store password. The exported certificate file name can be **<hostname>-certificate.der**

10. Restart the server so that it can load the wallet file and key store contents. It is important to restart the server after this step because WebLogic maintains a cache of these files which get refreshed when you restart the server.

11. Run the following command to update decorator jars with security policy configurations.

    ```
    <rsb-home>/service-assembly-home/bin
    ```

    ```
    rsb-compiler.sh -setup-security-credential
    ```

    It will prompt for sidb-user-alias-name, provide soainfra schema name.

    For example: rsb_soainfra

12. Prepare instrumentation configurations for WebLogic server.

    ```
    cd <rsb-home>/deployment-home/bin
    ```

    ```
    rsb-deployer.sh -prepare-wls
    ```

How to configure OWSM to use KSS Keystore in RSB domain

The keystore contains the entities private keys and the certificates associated with those private keys.

Using the OPSS Keystore Service for Message Protection:

In previous releases of OWSM, the JKS keystore was used by default. As of version 12.1.2.2, the OPSS Keystore Service is used by default for original installations. If you are upgrading from a prior release, your existing JKS keystore is used.

Migrating a JKS Keystore into the KSS Keystore

If you have an existing JKS keystore, you can migrate one or more aliases from the JKS keystore to the KSS keystore. To do this, perform the following steps:

Make sure the aliases you want to import are in the JKS keystore. You can use keytool (or your tool of choice) to do this

Keytool -list -keystore <keystore_name>.jks

For example: Keytool -list -keystore <hostname>-keystore.jks

Use WLST to import one or more aliases from the JKS keystore using the importKeyStore script at the command line.

1. Navigate to <WLS_HOME>/wlserver/common/bin

2. Run wlst.sh command, the prompt appears as **wls:/offline>**

3. Then connect to weblogic admin server using **connect("username", "password", "t3://<hostname>:<port>")**

   For example: connect("weblogic<username>", "weblogic1<password>", "t3://rtgdev03.in.oracle.com<host name>:38001<port>")

   Now it will connect to AdminServer as: **wls:/<domain_name>/serverConfig>**

   The starting point for using the Keystore Service command set is **getOpssService**, which gets an OPSS service command object that enables you to:

   - execute commands for the service
   - obtain command help

   The general syntax is:

Variable = getOpssService(name = 'service_name')

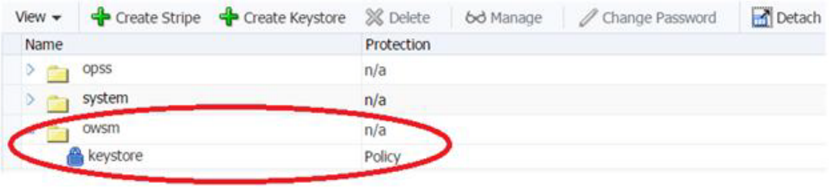
Where

- the variable stores the command object
- the service name refers to the service whose command object is to be obtained. The only valid value is 'KeyStoreService'.

4. svc=getOpssService(name='KeyStoreService')

   - Creates a new keystore on the given application stripe.

5. svc.createKeyStore(appStripe='owsm', name='keystore', password='<password>', permission=true)

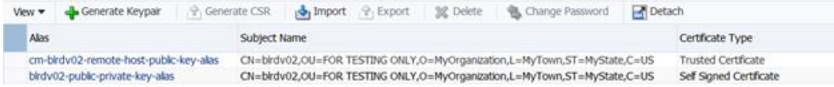


Name	Protection
opss	n/a
system	n/a
owsm	n/a
keystore	Policy

   - Imports a keystore from a specified file.

   importKeyStore(appStripe='stripe', name='keystore', password='<password>', aliases='comma-separated-aliases', keypasswords='comma-separated-keypasswords', type='keystore-type', permission=true | false, filepath='absolute_file_path')

6. svc.importKeyStore(appStripe='owsm',name='keystore',password='<password>',aliases='<Hostname>-public-private-key-alias,<appName>-<Hostname>-remote-host-public-key-alias',keypasswords='<password1>,<password2>',type='JKS',permission=true,filepath=' /<domain_home>/config/<Hostname>-keystore.jks')

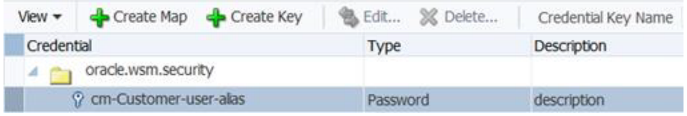


Alias	Subject Name	Certificate Type
cm-blrdv02-remote-host-public-key-alias	CN=blrdv02,OU=FOR TESTING ONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US	Trusted Certificate
blrdv02-public-private-key-alias	CN=blrdv02,OU=FOR TESTING ONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US	Self Signed Certificate

   - OWSM uses the Credential Store Framework (CSF) to manage the credentials in a secure form. The CSF provides a way to store, retrieve, and delete credentials for a Web Service and other applications.
   - Create a credential with key "<appName>-<ServiceName>-user-alias" in "oracle.wsm.security" map. The values must match with the username/password configured in application service's WebLogic server for UsernameToken authentication.

7. createCred(map="oracle.wsm.security", key="<appName>-<ServiceName>-user-alias", user="<application service user>", password="<application service user pwd>", desc="description")

   createCred(map="oracle.wsm.security", key=" cm-Customer-user-alias", user="<username>", password="<password>", desc="description")



Credential	Type	Description
oracle.wsm.security		
cm-Customer-user-alias	Password	description

   - OWSM provides support for KSS, JKS, HSM, and PKCS11 keystores. After creating the keystores, we need to configure OWSM so that it can access and use the keystore. We can configure the OWSM keystore using the configureWSMKeystore command.

- Sets the configuration properties for the OWSM keystore.

8. configureWSMKeystore(context, keystoreType,location, keystorePassword, signAlias, signAliasPassword, cryptAlias, cryptAliasPassword)

   configureWSMKeystore('/wls/<domain_ name>',keystoreType='KSS',location='kss://owsm/keystore',signAlias='<sign alias>,cryptAlias=<crypt alias>')

   - Run the following command to deploy the decorator:

     `cd <rsb-home>/deployment-home/bin`

     `rsb-deployer.sh -deploy-all-rsb-service-for-app <app>`

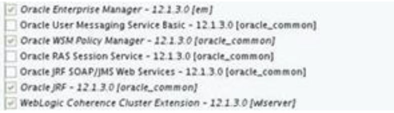
     (For example: **`rsb-deployer.sh -deploy-all-rsb-service-for-app sim`**)

   - Create user manually from admin console.
   - Restart the server/servers.

Configuring Consumer Side (Policy B)

The consumer should be deployed in a JRF domain with the following templates:

- Oracle EM
- Oracle WSM Policy Manager
- Oracle JRF
- Weblogic Coherence Cluster Extension



☑ Oracle Enterprise Manager - 12.1.3.0 [em]
□ Oracle User Messaging Service Basic - 12.1.3.0 [oracle_common]
☑ Oracle WSM Policy Manager - 12.1.3.0 [oracle_common]
□ Oracle RAS Session Service - 12.1.3.0 [oracle_common]
□ Oracle JRF SOAP/JMS Web Services - 12.1.3.0 [oracle_common]
☑ Oracle JRF - 12.1.3.0 [oracle_common]
☑ WebLogic Coherence Cluster Extension - 12.1.3.0 [wlserver]

This section provides details on setting up security in the service-consumer side of the RSB application.

Service consumer application must be run inside the WebLogic server. To consume services secured with Policy B, the application accesses the key store from the <DomainHome>/config folder of the WebLogic server where it is deployed. The application must also provide the username and password to authenticate the user with the service. Oracle recommends that you save the user credentials in a wallet file that is accessible from the consumer application. For simplicity purposes, the sample code provided below uses a hard coded username and password.

Note: For a wallet file, access the CSM package that Oracle provides.

Note: In the steps where keystore is created, it is important to note that the naming convention for alias and password are followed as provided in the samples below. The naming conventions that you can use are provided below.

1. The key store name must be **`<hostname>-keystore.jks`**.
2. The alias name for the server's public/private key must be **`<hostname>-public-private-key-alias`**.
3. The alias name of remote RSB/OSB server's when imported in the key store must be **`<remoteHostname>-remote-host-public-key-alias`**.

   The same naming convention must be followed in the service consumer code for calling remote OSB services.

Take the following steps to set up the keystore and call the service:

1. Copy **`security-setup-home/service-consumer/generate-pki-certificate-keystore.sh`** and **`import-remote-osb-server-public-key-certificate-into-keystore.sh`** from the **`RsbAppServiceSecuritySetupSamplesPak16.0.027ForAll16.0.027Apps_eng_ga.zip`** to the **`<domain-home>/config`** directory of the WebLogic application server where the service consumer application would be deployed.

   Navigate to **`<domain-home>/bin`** and set domain

   ```
   . setDomainenv.sh
   ```

2. Generate the certificate and the public/private key for the WebLogic server where the client application is deployed. Run the script located in security-setup-home/service-consumer. Navigate to **`<domain-home>/config`** and run.

   ```
   generate-pki-certificate-keystore.sh
   ```

3. Copy the certificate from the OSB server to the WebLogic server of the client application and run the script located in security-setup-home/service-consumer.

   ```
   import-remote-osb-server-public-key-certificate-into-keystore.sh
   <remoteHostName>
   ```

   Note down the key store password and private key password. The aliases used for the certificates and keys are fixed in this script. In your client code, you will have to use the following values for the program variables in the sample code mentioned in Appendix B.

Table 3–1 Configuring Consumer Slide

Program Variable	Value
clientKeyStore	<DomainHome>/config/<HostName>-keystore.jks
clientKeyStorePass	Keystore password you entered in this step 2
clientKeyAlias	<HostName>-public-private-key-alias
clientKeyPass	Private key password you entered in this step 2
remoteHostAlias	<RemoteHostName>-remote-host-public-key-alias
userNameTokenProvider	The username and the password for the Web service provider.

4. Export the certificate from <Hostname>-keystore.jks to the <domain-home>/config directory. This certificate path need to mention in the consumer side sample code. Copy the script app-service-security-config.sh from /security-setup-home/service-provider/config and run

```
app-service-security-config.sh -export-cert
```

5. The client also needs to provide a username and password for authentication. Create username and password for the web service provider using WebLogic Admin console. Oracle recommends that you store the username and password in a wallet file and read from there in the client code. In the sample code, the username and password have been hard-coded for simplicity.

> **Note:** For a wallet file, access the CSM package that Oracle provides.

6. When the OSB services are consumed from an application, the application could be running in a WebLogic server where the OWSM is also installed. Alternatively, it can also be consumed from inside a WebLogic server where it is only a base WebLogic server and there is no OWSM component installed. The approach differs based on the situation.

   In the WebLogic server where the OWSM is not available, the client code uses the WebLogic policies in the client side. A sample program is available in Appendix B.

   When the client application is running on a WebLogic server where the OWSM is available, the OWSM client policy can be used in client code. A sample program is available in Appendix C. The security related code is shown in ***bold-italics***.

How to configure OWSM to use KSS Keystore in Consumer Domain

The keystore contains the entities private keys and the certificates associated with those private keys.

Using the OPSS Keystore Service for Message Protection:

In previous releases of OWSM, the JKS keystore was used by default. As of version 12.1.2, the OPSS Keystore Service is used by default for original installations. If you are upgrading from a prior release, your existing JKS keystore is used.

Migrating a JKS Keystore Into the KSS Keystore

If you have an existing JKS keystore, you can migrate one or more aliases from the JKS keystore to the KSS keystore. To do this, perform the following steps:

Make sure the aliases you want to import are in the JKS keystore. You can use keytool (or your tool of choice) to do this

1. keytool -list -keystore <keystore_name>.jks

   For example: Keytool -list -keystore <hostname>-keystore.jks

   Use WLST to import one or more aliases from the JKS keystore using the importKeyStore script at the command line.

2. Go to <WLS_HOME>/wlserver/common/bin

3. Run wlst.sh command, the prompt appears as **wls:/offline>**

4. Then connect to weblogic admin server using **connect("username", "password", "t3://<hostname>:<port>")**

   Now it will connect to AdminServer as: **wls:/<domain_name>/serverConfig>**

   The starting point for using the Keystore Service command set is **getOpssService**, which gets an OPSS service command object that enables you to:

- execute commands for the service
- obtain command help

The general syntax is:

Variable = getOpssService(name = 'service_name')

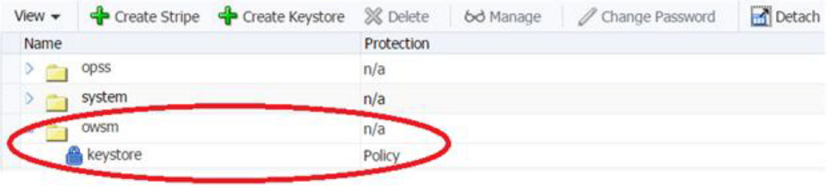
Where

- the variable stores the command object
- the service name refers to the service whose command object is to be obtained. The only valid value is 'KeyStoreService'.

5. svc=getOpssService(name='KeyStoreService')

   - Creates a new keystore on the given application stripe.

6. svc.createKeyStore(appStripe='owsm', name='keystore', password='<password>', permission=true)

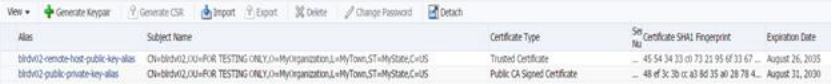


Name	Protection
opss	n/a
system	n/a
owsm	n/a
keystore	Policy

   - Imports a keystore from a specified file.

     importKeyStore(appStripe='stripe', name='keystore', password='<password>', aliases='comma-separated-aliases', keypasswords='comma-separated-keypasswords', type='keystore-type', permission=true | false, filepath='absolute_file_path')

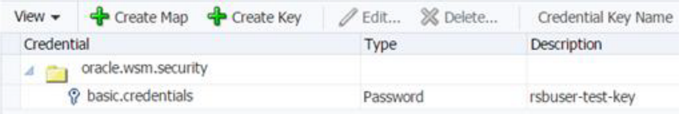
7. svc.importKeyStore(appStripe='owsm',name='keystore',password='<password>',aliases='<Hostname>-public-private-key-alias,<Hostname>-remote-host-public-key-alias',keypasswords='<password1>,<password2>',type='JKS',permission=true,filepath=' /<domain_home>/config/<Hostname>-keystore.jks')



Alias	Subject Name	Certificate Type	Se Nu	Certificate SHA1 Fingerprint	Expiration Date
blrdvi02-remote-host-public-key-alias	CN=blrdvi02,OU=FOR TESTING ONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US	Trusted Certificate	...	45 54 34 33 c0 73 21 95 6f 33 67 ...	August 26, 2035
blrdvi02-public-private-key-alias	CN=blrdvi02,OU=FOR TESTING ONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US	Public CA Signed Certificate	...	48 ef 3c 3b cc a3 8d 35 a0 28 78 4...	August 31, 2030

   - Creates a new credential in the domain credential store with a given map name, key name, user name, password and description. A credential store is the repository of security data that certifies the authority of entities used by Java 2, Java EE, and ADF applications. Applications can use the Credential Store, a single, consolidated service provider to store and manage their credentials securely. Use map name and key name as specified.

8. createCred(map="oracle.wsm.security", key="basic.credentials", user="<application service user>", password="<application service user pwd>", desc="description")



Credential	Type	Description
oracle.wsm.security		
basic.credentials	Password	rsbuser-test-key

- OWSM provides support for KSS keystore. After creating the keystore, we need to configure OWSM so that it can access and use the keystore. We can configure the OWSM keystore using the configureWSMKeystore command.
- Sets the configuration properties for the OWSM keystore.

  configureWSMKeystore(context, keystoreType,location, keystorePassword, signAlias, signAliasPassword, cryptAlias, cryptAliasPassword)

9. configureWSMKeystore('/wls/<domain_name>',keystoreType='KSS',location='kss://owsm/keystore',signAlias='<sign alias>,cryptAlias=<crypt alias>')

   For example,

   configureWSMKeystore('/wls/<jrf_domain>',keystoreType='KSS',location='kss://owsm/keystore',signAlias='<Hostname>-public-private-key-alias',cryptAlias='<Hostname>-public-private-key-alias')

   - Restart the server/servers.

Policy B Consumer in WebLogic Server

If the Policy B consumer is running in the WebLogic server, it can use the WebLogic Web service configuration for the policy related processing. The consumer code need not specify the keystore location, aliases etc. In order to utilize the container's security configuration, the WebLogic server must be configured for it. This can be done by running the following command in the <domain-home>/config directory.

Note: The script must be run after running the **setDomainEnv.sh** in the current shell.

```
$JAVA_HOME/bin/java -classpath $CLASSPATH weblogic.WLST configWss.py
```

The RIB-OMS application is an example of such usage.

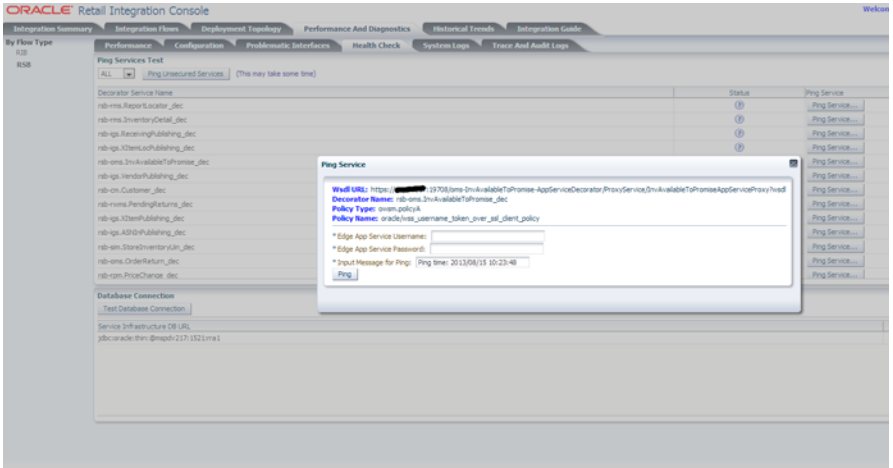
Post Installation Steps

After the successful configuration of security, verify it before running the functional tests. You can verify security by invoking a ping operation of a RSB decorator service securely.

It can be done in the following two ways:

1. Using Retail Integration Console (RIC)
2. Using the OSB Console

Verifying Policy Using the Retail Integration Console



Decorator Service Name
rsb-rms.ReportLocator_dec
rsb-rms.InventoryDetail_dec
rsb-igs.ReceivingPublishing_dec
rsb-igs.XItemLocPublishing_dec
rsb-oms.InvAvailableToPromise_dec
rsb-igs.VendorPublishing_dec
rsb-cm.Customer_dec
rsb-rwms.PendingReturns_dec
rsb-igs.XItemPublishing_dec
rsb-igs.ASNInPublishing_dec
rsb-sim.StoreInventoryUin_dec
rsb-oms.OrderReturn_dec
rsb-rpm.PriceChange_dec

Ping Service

Wsdl URL: https://[redacted]:19708/oms-InvAvailableToPromise-AppServiceDecorator/ProxyService/InvAvailableToPromiseAppServiceProxy?wsdl
Decorator Name: rsb-oms.InvAvailableToPromise_dec
Policy Type: owsm.policyA
Policy Name: oracle/wss_username_token_over_ssl_client_policy

* Edge App Service Username:
* Edge App Service Password:
* Input Message for Ping: Ping time: 2013/08/15 10:23:48

jdbc:oracle:thin:@mspdv217:1521:ma1

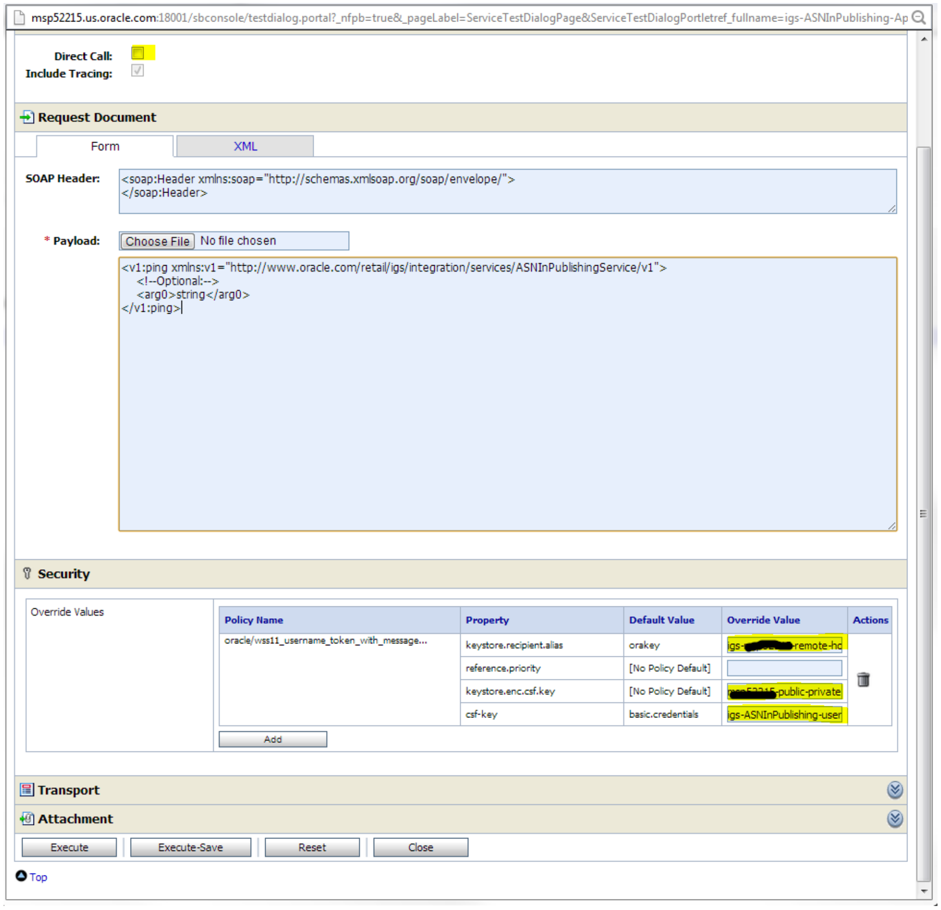
Verifying Policy Using the OSB Console

Click on the proxy service and then the Test Launch Console icon to test the service. You can invoke using any method. The ping method is the easiest to test. If the invocation is successful, it shows both the encrypted and decrypted versions of the request and response messages.

Make sure you do not select the **Direct Calls** check box. To test the proxy service from test page, provide values for following fields which are at the bottom of the test page:

- keystore.recipient.alias
- keystore.enc.csf.key
- csf-key

You can find the values for these fields in the Business Services --> <Business Service> --> **Security** tab of the same decorator project. You will see these values pre-populated in that page. Copy these values and enter them on the proxy service test page for a successful invocation of the service.



msp52215.us.oracle.com:18001/sbconsole/testdialog.portal?_nfpb=true&_pageLabel=ServiceTestDialogPage&ServiceTestDialogPortletref_fullname=igs-ASNInPublishing-Ap

Direct Call: □
Include Tracing: ☑

Request Document

SOAP Header:

```
<soap:Header xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
</soap:Header>
```

* Payload: Choose File No file chosen

```
<v1:ping xmlns:v1="http://www.oracle.com/retail/igs/integration/services/ASNInPublishingService/v1">
   <!--Optional:-->
   <arg0>string</arg0>
</v1:ping>
```

Security

Override Values

Policy Name	Property	Default Value	Override Value	Actions
oracle/wss11_username_token_with_message...	keystore.recipient.alias	orakey	igs-[redacted]-remote-hc	
	reference.priority	[No Policy Default]		
	keystore.enc.csf.key	[No Policy Default]	[redacted]-public-private	
	csf-key	basic.credentials	igs-ASNInPublishing-user	

Verifying Policy Using the Java Code

A sample Java code is shown in Appendix B that shows how a Java application can call a service secured with Policy B.

4 Troubleshooting

This chapter discusses common errors that can occur while implementing security and the corresponding resolution.

Error Getting Response; java.net.SocketException: Connection Reset

Description

The client application does not receive a response from the Web service. The error above is shown in SOAP User Interface (UI).

Solution

Check that the service is up and running, and that your client is pointing to the correct URL and using the correct protocol (HTTP or HTTPS).

SOAP Response is “Unknown Exception, Internal System Processing Error”

SOAP Response

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
   <env:Body>
  <env:Fault>
         <faultcode>env:Server</faultcode>
     <faultstring>Unknown exception, internal system processing
error.</faultstring>
  </env:Fault>
   </env:Body>
</env:Envelope>
```

Solution

There are one or more problems with your security settings. With policy A:

- Make sure the credentials are correct
- Make sure the WSS-Password Type = PasswordText and WSS Time To Live = 5000

- Make sure the time on your PC or client machine matches the time on the server machine

SOAP Response is “Error on Verifying Message Against Security Policy Error Code: 1025"

SOAP Response

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
   <env:Body>
  <env:Fault
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-sec
ext-1.0.xsd">
         <faultcode>wsse:InvalidSecurity</faultcode>
     <faultstring>Error on verifying message against security policy Error
code:1025</faultstring>
      </env:Fault>
   </env:Body>
</env:Envelope>
```

Solution

Make sure that the policy configured in WLS matches the tokens that the client (SOAPUI, Java application, WebInspect etc) is sending. For Policy A, you need to configure credentials, WSS Password Type = PasswordText and WSS Time to Live = 5000.

Null Pointer Exception

If the following exception is seen in server logs, it is generally because the server is not able to access the keystore. Either the keystore is corrupt or the password provided in domain’s Web service security configuration does not match with the actual password of the keystore.

```
java.lang.NullPointerException
at
weblogic.wsee.security.bst.ServerBSTCredentialProvider.getX509Credential(ServerBST
CredentialProvider.java:180)at
weblogic.wsee.security.bst.ServerBSTCredentialProvider.initCredentialFromContext(S
erverBSTCredentialProvider.java:161)at
weblogic.wsee.security.bst.ServerBSTCredentialProvider.initCredentials(ServerBSTCr
edentialProvider.java:111)at
weblogic.wsee.security.configuration.WssConfiguration.initDefaultConfiguration(Wss
Configuration.java:416)at
weblogic.wsee.security.configuration.WssConfiguration.init(WssConfiguration.java:1
34)at ....
```

Solution

Run the following script to configure WebLogic server.

```
app-service-security-config.sh -config-wls-cert-keystore
```

This script asks for the keystore and alias passwords. Enter the correct values. After the script is run, bounce the Weblogic server.

Useful Commands for Troubleshooting Security Issues

JVM Parameters for SSL Debug

```
-Dssl.debug=true -Dweblogic.security.SSL.ignoreHostnameVerification=true
-Dweblogic.security.SSL.enforceConstraints=off -Dweblogic.StdoutDebugEnabled=true
```

Location Identity and Trust Stores for WebLogic

```
/wlserver_12.2.1.3.0/server/lib/DemoIdentity.jks
/wlserver_12.2.1.3.0/server/lib/DemoTrust.jks
```

Generate a Certificate for Development Purposes

```
. <domain-home>/bin/setDomainEnv.sh
java utils.CertGen -certfile ServerCert -keyfile ServerKey -keyfilepass
DemoIdentityPassPhrase -cn rsbhost.example.com
```

Import the Certificate to DemoIdentity Keystore

```
java utils.ImportPrivateKey -certfile ServerCert.der -keyfile ServerKey.der
-keyfilepass DemoIdentityPassPhrase -keystore DemoIdentity.jks -storepass
DemoIdentityKeyStorePassPhrase -alias DemoIdentity -keypass DemoIdentityPassPhrase
```

Keytool Commands

```
keytool -list -v -keystore DemoIdentity.jks
keytool -delete -alias demoidentity -keystore DemoIdentity.jks
keytool -import -noprompt -trustcacerts -alias <AliasName> -file <certificate>
-keystore <KeystoreFile> -storepass <Password>
keytool -exportcert -storetype JKS -alias <AliasName>  -keystore <KeystoreFile>
-storepass <Password> -rfc -file <Certificate File Name>
```

Location of Java Keystore

```
%JAVA_HOME%\lib\security\cacerts
```

5

Security Considerations for Developers

Implementing security at an enterprise level involves making effective trade-offs and integrating security throughout your software development life cycle. One of the most effective ways to deal with security is to leverage proven principles, patterns, and practices. The key is to know which principles, patterns, and practices are effective for your particular situation.

Designing a Web service with security in mind presents an interesting set of challenges. Some are unique to service-oriented architecture and some are similar to the challenges that enterprise Web application development teams face.

A Web service is most commonly implemented as a wrapper; that is, as an interface between a consuming service and the back-end business logic components doing the actual work. A Web service acts as a trust boundary in your application architecture. By nature, a Web service acts as a gateway between the trusted business components and the less trusted or untrusted client components. For this reason, it is impossible to think about the security of a Web service without thinking about authentication, authorization, protection of sensitive data on the network, and handling potentially malicious input. Each of these areas represents key decisions you need to make in order to maintain the security of your application.

By following security best practices during the design of your Web service, you can use proven practices to improve your decision-making capabilities and make a cascading positive impact on the overall security of your application. Use the following design guidelines to reduce wasted effort trying to solve security problems for which there are already best practices in place to improve the security of your service:

- Understand the difference between threats, attacks, vulnerabilities, and counter measures.
- Understand the key distinctions for Service-Oriented Architecture (SOA).
- Understand the Web services security framework.
- Understand the key principles and patterns for building secure services.

Oracle Retail Web Service Security General Principles

Oracle Retail, at an enterprise level, does not try to dictate customers about the security policy they must use in their enterprise. Oracle Retail does provide security recommendations for its applications.

The RSB services security principles must be able to fit into your predefined corporate security policies / guidelines. Oracle Retail applications test, certify, and document a few commonly used security policies that work out of the box with minimal effort.

You are free to reconfigure to a higher or lower level of Web service security than that certified by Oracle Retail. In such situations, Oracle Retail does not provide setup/configuration documentations. You should follow the Fusion Middleware documentation and set up the system on your own. The security configuration is supported (as long as Fusion Middleware supports it) but not certified by Oracle Retail.

Technical Guidelines and Standards

Consider the following technical guidelines and standards while implementing security:

- All Web services provided by Oracle Retail must advertise their security policy in their WSDL file.
- Standard WS-Security 1.1 with WS-SecurityPolicy 1.2 assertions must be used in the WSDL files.
- Retail applications running on WebLogic server must use the newer WebLogic policy files. These files are prefixed with Wssp1.2-2007-.
- Avoid building custom security policy files and use out-of-the-box policies available in WebLogic server and OWSM.
- Authorization and data security is not within the scope of Web service security. Any such security requirement must be fulfilled by the individual applications.

A

Sample Java Policy A Client Program

```
package com.oracle.retail.oms.integration.client;
import java.net.URL;
import java.util.Map;
import javax.xml.namespace.QName;
import javax.xml.ws.BindingProvider;

import com.oracle.retail.integration.base.bo.custorderref.v1.CustOrderRef;
import com.oracle.retail.integration.base.bo.nothing.v1.Nothing;
import
com.oracle.retail.oms.integration.services.customerorderservice.v1.CustomerOrderPo
rtType;
import
com.oracle.retail.oms.integration.services.customerorderservice.v1.CustomerOrderSe
rvice;

import java.util.ArrayList;
import java.util.List;

import weblogic.wsee.security.unt.ClientUNTCredentialProvider;
import weblogic.xml.crypto.wss.WSSecurityContext;
import weblogic.xml.crypto.wss.provider.CredentialProvider;
import weblogic.wsee.jws.jaxws.owsm.SecurityPoliciesFeature;

public class CustomerClient {
 private void invokeHttpsService(){
  try {
   QName qName = new
QName("http://www.oracle.com/retail/oms/integration/services/CustomerOrderService/
v1",
 "CustomerOrderService");
   URL url = new
URL("https://rsbhost:22004/oms-CustomerOrder-AppServiceDecorator/ProxyService/Cust
omerOrderAppServiceProxy?wsdl");
   CustomerOrderService myService = new CustomerOrderService(url, qName);
   String[] policyA = new String[] { "policy:oracle/wss_username_token_over_ssl_
client_policy" };
SecurityPoliciesFeature securityFeatures = new SecurityPoliciesFeature(policyA);
CustomerOrderPortType myServicePort =
myService.getPort(CustomerOrderPortType.class, securityFeatures);
   List<CredentialProvider> credProviders =
             new ArrayList<CredentialProvider>();
   // These should come from wallet in production code.
   String username = "rsbuser";
   String password = <rsbuser password>;
   CredentialProvider cp =
         new ClientUNTCredentialProvider(username.getBytes(),
                      password.getBytes());
   credProviders.add(cp);
   BindingProvider bindingProvider = (BindingProvider) myServicePort;
   Map<String, Object> requestContext = bindingProvider.getRequestContext();
   requestContext.put(WSSecurityContext.CREDENTIAL_PROVIDER_LIST,
                       credProviders);
   requestContext.put(BindingProvider.USERNAME_PROPERTY, username);
   requestContext.put(BindingProvider.PASSWORD_PROPERTY, password);
   Nothing nothing = new Nothing();
   nothing.setDescription("nothing");
   CustOrderRef d =  myServicePort.requestNewCustomerOrderId(nothing);
   System.out.println("Got Response : " + d.getOrderId());
  } catch(Exception e) {
e.printStackTrace();
  }
 }
}
```

B

Sample Java Policy B Consumer Using WebLogic Policy

```
package com.test;
import weblogic.security.SSL.TrustManager;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.security.KeyStore;
import java.security.cert.X509Certificate;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.namespace.QName;
import javax.xml.ws.BindingProvider;

import weblogic.jws.jaxws.ClientPolicyFeature;
import weblogic.jws.jaxws.policy.InputStreamPolicySource;
import weblogic.xml.crypto.wss.WSSecurityContext;
import weblogic.xml.crypto.wss.provider.CredentialProvider;
import weblogic.wsee.security.bst.ClientBSTCredentialProvider;
import weblogic.wsee.security.unt.ClientUNTCredentialProvider;

import
com.oracle.retail.cm.integration.services.customerservice.v1.CustomerPortType;
import
com.oracle.retail.cm.integration.services.customerservice.v1.CustomerService;
import com.oracle.retail.integration.base.bo.customerref.v1.CustomerRef;
import
com.oracle.retail.integration.base.bo.invocationsuccess.v1.InvocationSuccess;

public class CustomerServlet extends HttpServlet {
 private static final long serialVersionUID = 5798562566372551297L;

 protected void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                     throws ServletException, IOException {
  doPost(request, response);
 }
 protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException {
  response.getWriter().write("<html><body> Policy B testing with WebLogic policy
</body></html>");

  try {
   // Key store information
   String hostName = getHostName();
   String clientKeyStore = "config/" + hostName + "-keystore.jks";
   String clientKeyStorePass = "123456"; // From wallet in production
   String clientKeyAlias = hostName + "-public-private-key-alias";
   String clientKeyPass = "123456";      // From wallet in production
   // Hostname for remote host alias can be found from the URL of the OSB service
   String remoteHostAlias = "rsbhost-remote-host-public-key-alias";

   // Get remote server certificate from key store
   FileInputStream fis = new FileInputStream(new File(clientKeyStore));
   KeyStore keystore = KeyStore.getInstance("JKS");
   keystore.load(fis, clientKeyStorePass.toCharArray());
   final X509Certificate serverCert =
           (X509Certificate) keystore.getCertificate(remoteHostAlias);
   serverCert.checkValidity();

   // Set security policies
   InputStream encryptBodyPolicy =
this.getClass().getClassLoader().getResourceAsStream("weblogic/wsee/policy/runtime
/Wssp1.2-2007-EncryptBody.xml");
   InputStream signBodyPolicy =
this.getClass().getClassLoader().getResourceAsStream("weblogic/wsee/policy/runtime
/Wssp1.2-2007-SignBody.xml");

   InputStream usernameTokenPolicy = this.getClass().getClassLoader()
.getResourceAsStream("weblogic/wsee/policy/runtime/Wssp1.2-2007-Wss1.1-UsernameTok
en-Plain-EncryptedKey-Basic128.xml");

   ClientPolicyFeature clientPolicyFeature = new ClientPolicyFeature();
   clientPolicyFeature.setEffectivePolicy(new
             InputStreamPolicySource(encryptBodyPolicy, signBodyPolicy,
                                     usernameTokenPolicy));

   // Prepare credential providers
   List<CredentialProvider> credProviders = new
            ArrayList<CredentialProvider>();
   CredentialProvider messageProtectionProvider = new
            ClientBSTCredentialProvider(clientKeyStore, clientKeyStorePass,
                clientKeyAlias, clientKeyPass, "JKS",      serverCert);
   credProviders.add(messageProtectionProvider);
   ClientUNTCredentialProvider userNameTokenProvider = new
         ClientUNTCredentialProvider("rsbuser".getBytes(),
                                     "rsbuser1".getBytes());
   credProviders.add(userNameTokenProvider);

   // Prepare service context
   String wsdlUrl =
"http://rsbhost:9004/cm-Customer-AppServiceDecorator/ProxyService/CustomerAppServi
ceProxy?wsdl";

   CustomerService service = new CustomerService(
           new URL(wsdlUrl),
           new QName(
"http://www.oracle.com/retail/cm/integration/services/CustomerService/v1",
"CustomerService"));

   CustomerPortType servicePort =
               service.getCustomerPort(clientPolicyFeature);

   Map<String, Object> reqContext =
               ((BindingProvider) servicePort).getRequestContext();
   reqContext.put(WSSecurityContext.CREDENTIAL_PROVIDER_LIST, credProviders);

   reqContext.put(WSSecurityContext.TRUST_MANAGER, new TrustManager() {
public boolean certificateCallback(X509Certificate[] chain,
                    int validateErr) {
 // Check that the server cert matches
 boolean result = chain[0].equals(serverCert);
 return result;
}
   });

   // Invoke the service
   CustomerRef customerRef = new CustomerRef();
   customerRef.setCustomerId("1");
   InvocationSuccess invSuccess = servicePort.deleteCustomer(customerRef);
   response.getWriter().write("<html><body>Got Response : "
           + invSuccess.getSuccessMessage() + "</body></html>");
  } catch (Exception e) {
e.printStackTrace();
  }
  response.getWriter().flush();
  response.getWriter().close();
 }

// This method returns the hostname of the server where application is running
   private String getHostName(){
String wlsHostName = null;
try {
 String hostName = java.net.InetAddress.getLocalHost().getHostName();
 wlsHostName = hostName.split("\\.")[0];
} catch (UnknownHostException e) {
  throw new RuntimeException(e);
}
return wlsHostName;
   }
}
```

C

Sample Java Policy B Consumer Using OWSM Client Policy

```
package com.test;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.net.UnknownHostException;
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SignatureException;
import java.security.UnrecoverableKeyException;
import java.security.cert.CertPath;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.namespace.QName;
import javax.xml.ws.BindingProvider;
import oracle.security.jps.JpsContext;
import oracle.security.jps.JpsContextFactory;
import oracle.security.jps.JpsException;
import oracle.security.jps.service.keystore.KeyStoreService;
import weblogic.jws.jaxws.ClientPolicyFeature;
import weblogic.jws.jaxws.policy.InputStreamPolicySource;
import weblogic.security.SSL.TrustManager;
import weblogic.wsee.jws.jaxws.owsm.SecurityPoliciesFeature;
import weblogic.wsee.security.bst.ClientBSTCredentialProvider;
import weblogic.wsee.security.unt.ClientUNTCredentialProvider;
import weblogic.xml.crypto.wss.WSSecurityContext;
import weblogic.xml.crypto.wss.provider.CredentialProvider;
import
com.oracle.retail.cm.integration.services.customerservice.v1.CustomerPortType;
```

```
import
com.oracle.retail.cm.integration.services.customerservice.v1.CustomerService;
import
com.oracle.retail.cm.integration.services.customerservice.v1.IllegalArgumentWSFaul
tException;
import
com.oracle.retail.cm.integration.services.customerservice.v1.IllegalStateWSFaultEx
ception;
import
com.oracle.retail.cm.integration.services.customerservice.v1.ValidationWSFaultExce
ption;
import com.oracle.retail.integration.base.bo.customerref.v1.CustomerRef;
import
com.oracle.retail.integration.base.bo.invocationsuccess.v1.InvocationSuccess;
/**
* Servlet implementation class PolicyBJRFServlet
*/
@WebServlet("/PolicyBJRFServlet")
public class PolicyBJRFServlet extends HttpServlet {
private static final long serialVersionUID = 1L;
/**
* @see HttpServlet#HttpServlet()
*/
public PolicyBJRFServlet() {
super();
// TODO Auto-generated constructor stub
}
/**
* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
*/
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
doPost(request, response);
}
/**
* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
*/
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// TODO Auto-generated method stub
response.getWriter().write("<html><body> Policy B testing with WebLogic policy
</body></html>");
try {
// Key store information
String hostName = getHostName();
//Point to consumer weblogic domain config
String str = "/home/dev02/Weblogic12.2./user_projects/domains/jrfb_
domain/config/";
String clientKeyAlias = hostName+"-public-private-key-alias";
// Hostname for remote host alias can be found from the URL of the OSB service
String remoteHostAlias = hostName+"-remote-host-public-key-alias";
JpsContext ctx = JpsContextFactory.getContextFactory().getContext();
KeyStoreService kss = ctx.getServiceInstance(KeyStoreService.class);
java.security.KeyStore keystore = kss.getKeyStore("owsm", "keystore",null);
Key key = keystore.getKey(clientKeyAlias, null);
// Get the certificate associated with this alias
final X509Certificate serverCert =
(X509Certificate) keystore.getCertificate(remoteHostAlias);
System.out.println("serverCert----------------"+serverCert.toString());
PublicKey publicKey = serverCert.getPublicKey();
// Set security policies
SecurityPoliciesFeature securityFeatures = new SecurityPoliciesFeature(
new String[] {
"oracle/wss11_username_token_with_message_protection_client_policy" });
// Prepare credential providers
List<CredentialProvider> credProviders = new
ArrayList<CredentialProvider>();
PrivateKey pk = (PrivateKey)key;
String fileName = str+hostName+"-certificate.der";
CredentialProvider messageProtectionProvider = new
ClientBSTCredentialProvider(serverCert, CertPathMeth("X.509", fileName), pk);
credProviders.add(messageProtectionProvider);
ClientUNTCredentialProvider userNameTokenProvider = new
ClientUNTCredentialProvider("rsbuser".getBytes(),
"rsbuser1".getBytes());
credProviders.add(userNameTokenProvider);
//point to osb proxy url with cluster port
String wsdlUrl =
"http://blrdv02.in.oracle.com:49004/cm-Customer-AppServiceDecorator/ProxyService/C
ustomerAppServiceProxy?wsdl";
//pass namespace and service name of the proxy service
CustomerService service = new CustomerService(
new URL(wsdlUrl),
new QName(
"http://www.oracle.com/retail/cm/integration/services/CustomerService/v1",
"CustomerService"));
CustomerPortType servicePort =
service.getCustomerPort(securityFeatures);
Map<String, Object> reqContext = ((BindingProvider)
servicePort).getRequestContext();
reqContext.put(WSSecurityContext.CREDENTIAL_PROVIDER_LIST, credProviders);
reqContext.put(WSSecurityContext.TRUST_MANAGER, new TrustManager() {
public boolean certificateCallback(X509Certificate[] chain,
int validateErr) {
// Check that the server cert matches
boolean result = chain[0].equals(serverCert);
return result;
}
});
// Invoke the service
CustomerRef customerRef = new CustomerRef();
customerRef.setCustomerId("3451");
InvocationSuccess invSuccess = servicePort.deleteCustomer(customerRef);
response.getWriter().write("");
response.getWriter().write("<html><body>Got Response : "
+ invSuccess.getSuccessMessage() + "</body></html>");
}catch (JpsException e2) {
e2.printStackTrace();
}
catch (UnrecoverableKeyException | KeyStoreException
| NoSuchAlgorithmException e2) {
e2.printStackTrace();
}
catch (IllegalArgumentWSFaultException | IllegalStateWSFaultException
| ValidationWSFaultException e) {
e.printStackTrace();
}
response.getWriter().flush();
response.getWriter().close();
}
private CertPath CertPathMeth(String string, String fileName) {
CertPath cp = null;
try {
CertificateFactory cf = CertificateFactory.getInstance(string);
List<java.security.cert.Certificate> list = new
ArrayList<java.security.cert.Certificate>();
InputStream in = new FileInputStream(fileName);
java.security.cert.Certificate c = cf.generateCertificate(in);
list.add(c);
cp = cf.generateCertPath(list);
System.out.println("Clientcert==============="+cp.getCertificates().get(0).toStrin
g());
} catch (java.io.FileNotFoundException e) {
e.printStackTrace();
}
catch (CertificateException e) {
e.printStackTrace();
}
return cp;
}
// This method returns the hostname of the server where application is running
private String getHostName(){
String wlsHostName = null;
try {
String hostName = java.net.InetAddress.getLocalHost().getHostName();
wlsHostName = hostName.split("\\.")[0];
} catch (UnknownHostException e) {
throw new RuntimeException(e);
}
return wlsHostName;
}
}
```

Note:

`retail-public-payload-java-beans-base-<version>.jar`, **`retail-public-payload-java-beans-<version>.jar`**, **`<service>ServiceConsumer.jar and javax.servlet-api-3.0.1.jar`** should be present in the classpath.