Oracle® Retail Integration Bus

Security Guide

Release 16.0.21

E87013-01

May 2017

ORACLE®

Oracle Retail Integration Bus Security Guide, Release 16.0.21

E87013-01

Primary Author: Sanal Parameswaran

Contributing Author: Maria Andrew

Contents

Send Us Your Comments

Oracle Retail Integration Bus Security Guide, Release 16.0.21

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

This document serves as a guide for administrators, developers, and system integrators who securely administer, customize, and integrate applications using Oracle Retail Integration Bus (RIB). Installation and configuration for each product are covered in more detail in the each product's Installation Guide.

Audience

This document is intended for administrators, developers, and system integrators who perform the following functions:

- Document specific security features and configuration details for the Oracle Retail Integration Bus products, in order to facilitate and support the secure operation of the Oracle Retail product and any external compliance standards.
- Guide administrators, developers, and system integrators on secure product implementation, integration, and administration.

It is assumed that the readers have general knowledge of administering the underlying technologies and the application.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in the Oracle Retail documentation set:

- *Oracle Retail Integration Bus Implementation Guide*
- *Oracle Retail Integration Bus Installation Guide*
- *Oracle Retail Integration Cloud Service Release Notes*

- *Oracle Retail Integration Bus Hospital Administration Guide*
- *Oracle Retail Integration Bus Operations Guide*
- *Oracle Retail Integration Bus Support Tools Guide*
- *Oracle Retail Integration Bus Java Messaging Service (JMS) Console Guide*
- *Oracle Retail Enterprise Integration Guide*
- *Oracle Retail Functional Artifacts Guide*
- *Oracle Retail Integration Bus Integration Gateway Services Guide*
- *Oracle Retail Functional Artifact Generator Guide*
- *Oracle Retail Service-Oriented Architecture Enabler Tool Guide*

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, 16.0.21). 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/technetwork/documentation/oracle-retail-100266.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

Security Overview

Security in the integration layer is a big concern for every retail enterprise. The security system should be open enough to allow trusted remote applications to integrate easily and, at the same time, lock down unauthorized remote access. To address security concerns, RIB utilizes the security modules available in the Oracle middleware and database systems.

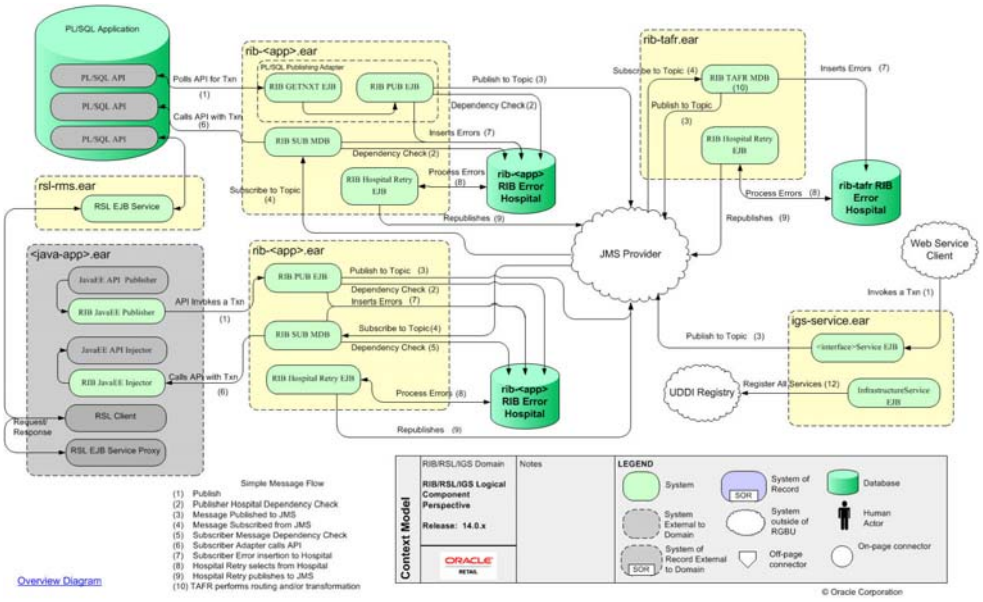
This chapter provides an overview of the security features in Oracle Retail Integration Bus. It includes the following sections:

- Physical Deployment Model
- Dependent Applications
- General Security Principles
- Recommended Approach

Physical Deployment Model

The following figure illustrates the physical deployment model of the RIB application.

Figure 1–1 Physical Deployment Model of RIB Application



The Web-based RIB Administrator user interface is accessed using a browser. You are responsible for applying the necessary security patches to the Web browser and the operating system.

The typical configuration of RIB runs on the following server:

- Oracle WebLogic Server 12c Release 2 (12.2.1.2) that hosts the rib application (rib-<app>.ear).
- Oracle Database 12c Release 1 (12.1.0.2).

You are responsible for applying any critical patch updates releases for the server hardware, application server, and the database.

Dependent Applications

Security Guides for dependent applications can be found at the following links.

- Oracle Database 12c Release 1(12.1.0.2) Enterprise Edition: http://docs.oracle.com/database/121/DBSEG/toc.htm.
- Oracle WebLogic Server 12c Release 2 (12.2.1.2):

  https://docs.oracle.com/middleware/1221/wls/INTRO/security.htm#INTRO232

Oracle Retail Integration Bus Administration User Interface

Each RIB application PAK ear i.e. rib-<app>.ear deployed in application server has a Web based user interface application which can be accessed via HTTP or HTTPS. If the HTTP protocol is disabled, and accessing this application over HTTP is not possible, a valid SSL certificate needs to be installed to access the administration GUI over HTTPS. To install a valid SSL Certificate on the application server, see the documentation for your installed application server. The use of the default SSL certificate shipped with the application server is not recommended because it renders the application prone to intrusion attacks.

General Security Principles

Security is fundamentally about protecting assets. It is important to recognize that security is a path, not a destination. As you analyze your infrastructure and applications, you identify potential threats and understand that each threat presents a degree of risk. 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.

The Foundations of Security

Security relies on the following elements:

- **Authentication**

  Authentication addresses the question: who are you? It is the process of uniquely identifying the clients of your applications and services. These might be end users, other services, processes, or computers. In security parlance, authenticated clients are referred to as principals.

- **Authorization**

  Authorization addresses the question: what can you do? It is the process that governs the resources and operations that the authenticated client is permitted to

access. Resources include files, databases, tables, rows, and so on, together with system-level resources such as registry keys and configuration data. Operations include performing transactions such as purchasing a product, transferring money from one account to another, or increasing a customer's credit rating.

- **Auditing**

  Effective auditing and logging is the key to non-repudiation. Non-repudiation guarantees that a user cannot deny performing an operation or initiating a transaction.

- **Confidentiality**

  Confidentiality, also referred to as privacy, is the process of making sure that data remains private and confidential, and that it cannot be viewed by unauthorized users or eavesdroppers who monitor the flow of traffic across a network. Encryption is frequently used to enforce confidentiality. Access control lists (ACLs) are another means of enforcing confidentiality.

- **Integrity**

  Integrity is the guarantee that data is protected from accidental or deliberate (malicious) modification. Like privacy, integrity is a key concern, particularly for data passed across networks. Integrity for data in transit is typically provided by using hashing techniques and message authentication codes.

Oracle Credential Store Framework API Principles

A credential store is used for secure storage of credentials. The Credential Store Framework (CSF) API is used to access and perform operations on the credential store. The Credential Store Framework:

- Enables you to manage credentials securely.
- Provides an API for storage, retrieval, and maintenance of credentials in different back-end repositories.
- Supports file-based (Oracle wallet) and LDAP-based credential management.

Critical (create, update, delete) functions provided by the CSF API include:

- Verifying if a credential map, or a credential with a given key, exists in the store.
- Returning credentials associated with <mapname, key>.
- Assigning credentials to <mapname, key>.
- Deleting credentials associated with a given map name, or a given map name and key.
- Resetting credentials for a specified <mapname, key>.

Oracle Retail Recommended Security Approach

This section discusses the two security approaches that Oracle Retail recommends.

Oracle Software Security Assurance (OSSA)

Encompassing every phase of the product development lifecycle, Oracle Software Security Assurance (OSSA) is Oracle's methodology for building security into the design, build, testing, and maintenance of its products. Oracle's goal is to ensure that

Oracle's products, as well as the customer systems that leverage those products, remain as secure as possible.

OSSA compliance

To be OSSA compliant, it is required to use the CSF API to store the passwords in Oracle wallet based files. RIB and other products use the CSF API through a Credential Store Manager utility. This utility provides methods that can store and retrieve credentials from a wallet based file. Internally this utility is using CSF API to manage the credentials.

2

RIB Secure Installation and Configuration

This chapter explains how to securely configure Oracle Retail Integration Bus applications and related tools. For installation instructions, see the Installation Guide that accompanies each product.

Security in RIB Application Builder

RIB Application Builder is a tool for building and deploying RIB applications on the WebLogic server. The rib-deployment-env-info.xml file is the single source of all values used in the RIB App Builder tools. It is the only (or should be the only) file that requires editing. The RIB installer gathers the appropriate values from you, constructs the file, and invokes the appropriate tools.

```
<aq-jms-server jms-server-id="jms1">
<jms-server-home>linux1@linux1:/home/oracle/oracle/product/12.1.0/db_
1</jms-server-home>
<jms-url>jdbc:oracle:thin:@linux1:1521/pdborcl</jms-url>
<jms-port>1521</jms-port>
<jms-user-alias>jms1_user-name-alias</jms-user-alias>
</aq-jms-server>
```

This file does not contain the username and password for connecting to the application server or the databases. Rather, it contains the alias for each user name/password combination. This alias refers to the user name/password stored in a secured wallet file. The wallet file is created when the user runs the application assembly tool during the RIB application building process.

The syntax for the application assembly command is as follows:

```
rib-app-compiler.sh -setup-security-credential
```

The argument, *-setup-security-credential*, must be used when running the rib-app-compiler for the first time. It prompts the user to enter all usernames and passwords required to install RIB components. It stores details as credentials in a wallet file inside the *rib-home/deployment-home/conf/security/* directory. The credentials are retrieved and used by the deployer script when installing RIB components.

Only the operating system user who created the wallet file with the RIB application assembly tool has read and write access to the file. Other users do not have permission to access the file. The file permissions are set up during the post-deployment phase for RIB applications.

For more information about RIB Application Builder, see the *Oracle Retail Integration Bus Operations Guide*.

Note: You can also change usernames and passwords for the RIB applications after deploying them. Refer to the section "setup-security-credential," under "RIB App Builder Tools" in the "Application Builder" chapter in Oracle Retail Integration Bus Operations Guide for information on changing RIB usernames and passwords after deployment.

Security in RIB Deployment Configuration File Editor

The RIB Deployment Configuration File Editor is an application used to configure the rib-deployment-env-info.xml file, following installation. It provides a user interface for adding, removing, and rearranging the elements of the RIB configuration.

This tool has fields for entering usernames and passwords required for connecting to application server and databases. Values entered in the password field in the tool are displayed as a series of asterisks (one for each character). The values entered in this field are stored in the secured wallet file in the rib-home/deployment-home/conf/security/ directory.

For information about the RIB Deployment Configuration File Editor, see the section "RIB Deployment Configuration File Editor," in the "Application Builder" chapter in the *Oracle Retail Integration Bus Operations Guide*.

Security During RIB Deployment Process

You can run the RIB application assembly tool to build RIB application ear files. The generated .ear files contain deployment descriptors for data sources used by RIB runtime to connect to the application database and the error hospital database. The deployment descriptors contain the username for accessing the database, but the passwords are not stored there. During the deployment process for the RIB application, the passwords are read from the wallet file and encrypted using a WebLogic utility. The encrypted passwords are added in a WebLogic deployment plan that is uploaded on the server along with the .ear file.

Security During RIB Runtime

During the runtime process, the RIB application must make calls to the JMX server. WebLogic instance username and password are required to make connections to the JMX server. This information is stored in a secured wallet file, the path to which is stored in the rib-system.properties file.

For information about the properties in rib-system.properties file, see the "rib-system.properties" section in the "Backend System Administration and Logging" chapter of the *Oracle Retail Integration Bus Operations Guide*.

Only the operating system user who created them has read and write access to the properties files created during the RIB application deployment process. Other users do not have permission to access the files. Permissions are granted during the post deployment phase for RIB applications.

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"
```

Security in RIB<app>

RIB follows a role-based authorization for allowing valid users to perform a defined set of operations from rib-admin-gui. RIB users should belong to one of the following WebLogic user groups to be a authenticated user:

- ribAdminGroup
- ribOperatorGroup
- ribMonitorGroup

The following table defines the authorization rule for each role supported in RIB.It defines which role can perform what operations from the rib-admin-gui:

Table 2–1 Security in RIB

Role Name	AdminRole	OperatorRole	MonitorRole
GroupName	**ribAdminGroup**	**ribOperatorGroup**	**ribMonitorGroup**
Start/Stop Adapters	Yes	Yes	No
Chaneg Log levels	Yes	Yes	No
View Logs	Yes	Yes	Yes

Security in RIHA

Oracle Retail Integration Bus Hospital Administration or RIB Hospital Administration (RIHA) is a tool to manage RIB messages in the RIB error hospital tables. It is a Web application that is deployable on the WebLogic server.

For more information on setting up security for RIHA, see the "Installation and Setup" section in the *Oracle Retail Integration Bus Hospital Administration Guide*.

Security in RDMT

The RIB Diagnostic and Monitoring Toolkit (RDMT) is a collection of command line tools for controlling and monitoring RIB applications. When used from within rib-home, RDMT loads configuration information from the rib-deployment-env-info.xml file. For username and password information, it reads the wallet file created during the RIB application assembly process.

For information about RDMT, see the "Diagnostic and Monitoring Tools" chapter in the *Oracle Retail Integration Bus Operations Guide*.

Security in PL/SQL Application API Stubs

The plsql-api-stub is an API simulator designed to act as though the RIB is connected to the application, but it can process specific status and other parameters from a "stubbed" application. This set of tools is designed to emulate those applications exposing PL/SQL APIs to RIB, such as RMS and RWMS. The tool reads and writes the username and password for connecting to the database in a secured wallet file.

Security in Integration Gateway Services

The RIB Integration Gateway Services (IGS) component is a set of standard Simple Object Access Protocol (SOAP) based Web services that provide access to the RIB infrastructure. These Web services are generated using the Oracle Retail Service Enabler Tool. They should be secured after being deployed.

For more information, see Chapter 3 - Secure IGS Web Services Using the Administration Console in this guide.

Secure Sockets Layer Configuration

Secure Sockets Layer (SSL) provides secure connections by allowing two applications connecting over a network to authenticate each other's identity and encrypting the data exchanged between the applications. Configuring SSL in WebLogic servers in production environments is recommended.

For more information, see the WebLogic documentation on configuring SSL in WebLogic 12c server at
https://docs.oracle.com/middleware/1221/wls/SECMG/ssl.htm#CIHBDHEG

Deployment of RIB applications over SSL protocol are supported by entering protocol values as https in deployment info xml file. After the applications are deployed, they run on the SSL protocol.

The following are high level steps for running the RIB in SSL environment:

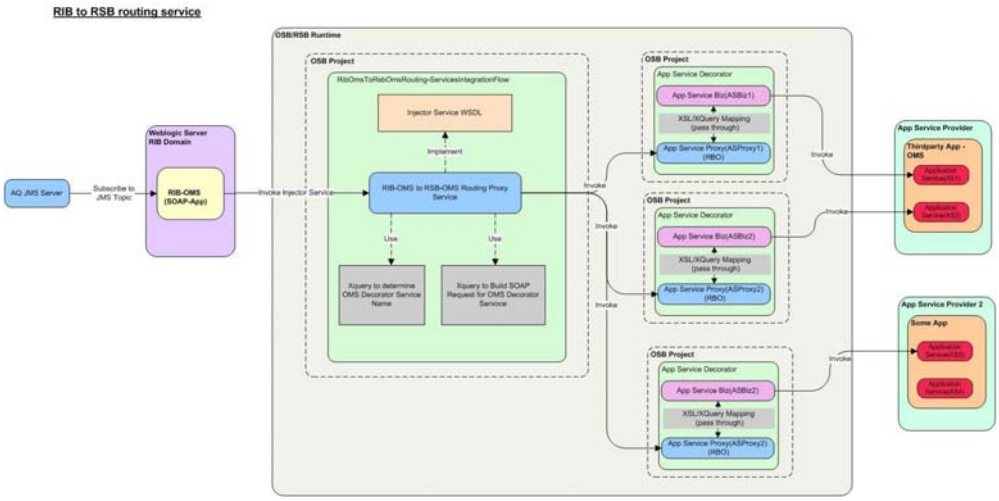
1. Configure SSL in the WebLogic server. (See WebLogic documentation for detailed steps.)
2. Keep the SSL ports of the WebLogic server instances open for RIB deployment. Verify that the SSL port is open: In the WebLogic administration console, go to the Configuration > General page of the server instance. Verify that the SSL Listen Port Enabled checkbox is checked and provide a unique listen address to each managed server and the admin server.
3. Make sure that the rib-deployment-env-info.xml file has protocol specified as https and port numbers are https port numbers for WebLogic server instances.
4. Start the managed server with SSL port enabled. For example, startManagedServer.sh rib-oms-server <AdminServerUrl>.
5. Deploy the RIB applications.
6. If required, non-SSL ports can be disabled as follows. In the WebLogic administration console, go to the **Configuration** > **General page** of the server instance. Uncheck the Listen Port Enabled checkbox and check the SSL Listen Port Enabled checkbox. This is an optional step and must be done only when all communications with the server are over HTTPS protocol.

Security in Injector Service

The RIB integrates with the Web service providers (example: OMS) using an injector service. This section contains details about the security configuration required on RIB side.

End application (OMS) Web services can be secured with policyA or policyB. For more details on application service security, see the *Oracle Retail Service Backbone Security Guide*.

Figure 2–1 RIB to RSB Routing Service



RIB-OMS Security Configuration

Provide the RibOmsToRsbOmsRoutingService URL, user-alias, and ws-policy-name information in rib-deployment-env-info.xml file present under rib-home/deployment-home/conf. The endpoint URL should be the secured injector service URL and user alias should be in <rib-app>_ws_security_user-name-alias format i.e. rib-oms_ws_security_user-name-alias. The ws-policy name should be either policyA or policyB depending on the OMS application Web service configuration.

Figure 2–2 RIB-OMS Security Configuration

```
<app id="oms" type="soap-app">
    <end-point>
        <url>https://blrqa01.idc.oracle.com:7505/RibOmsToRsbOmsRouting-ServicesIntegrationFlow/ProxyService/RibOmsToRsbOmsRoutingService</url>
        <!--  Do we want this location or derive by default? -->
        <!--  TODO : We need pick the ws policy from this location and push the policy to server. -->
        <!--<ws-policy-location>X/Y/Z</ws-policy-location>-->
        <ws-policy-name></ws-policy-name>
        <user-alias>rib-oms_ws_security_user-name-alias</user-alias>
    </end-point>
</app>
```

For more information on installing the RIB-OMS application, see the *Oracle Retail Integration Bus Installation Guide*.

RIB-OMS to RSB-OMS Routing Service Security Configuration

Security policies needs to be applied on each layer listed below:

- Application Web services secured with policyA or policyB.
- Decorators, which are proxy to actual application service, will also have security policies applied.

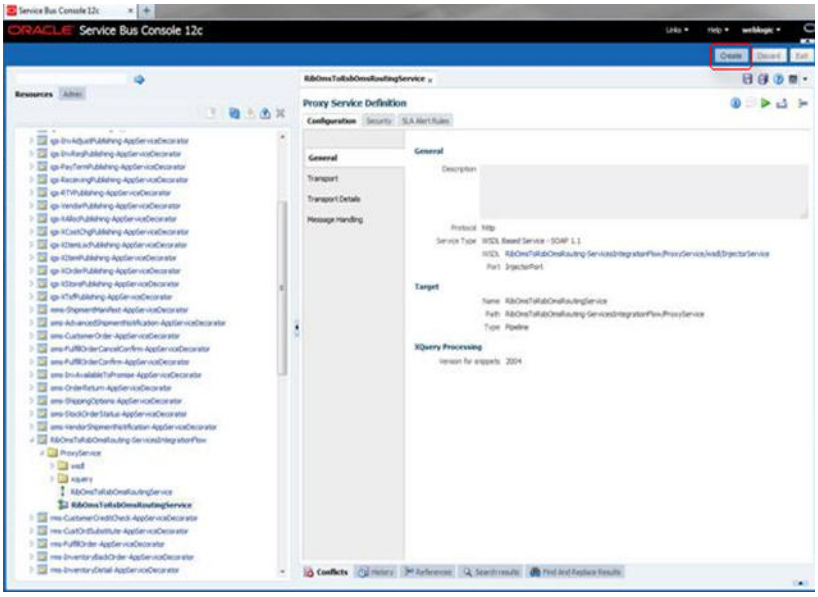
- Injector service which is a bridge between the RIB and the RSB will also have security policies applied.

For more information on applying security policies to decorator services, see *Oracle Retail Service Backbone Security Guide*.

Take the following steps to apply security policies in a routing service:

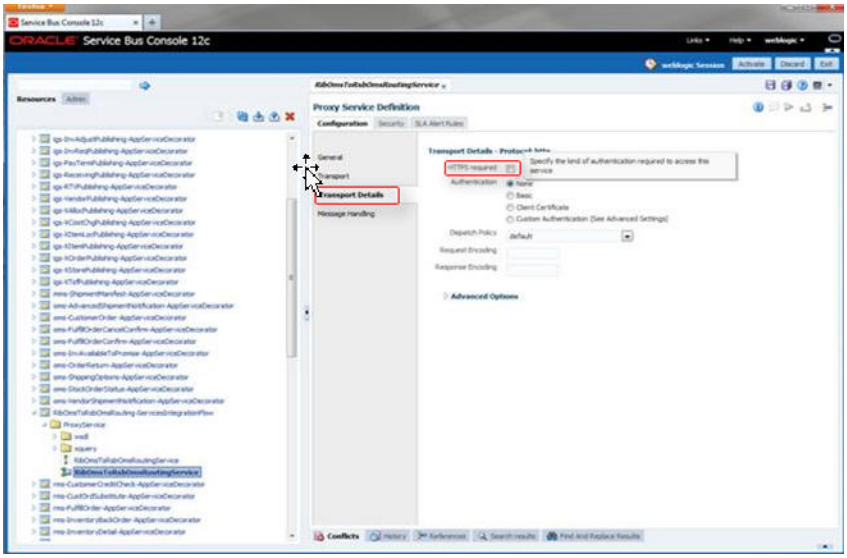
1. Click the routing service proxy *RibOmsToRsbOmsRoutingService*. The following window appears. Click **Create** button on Service Bus Console.

Figure 2–3 Edit a Proxy Service



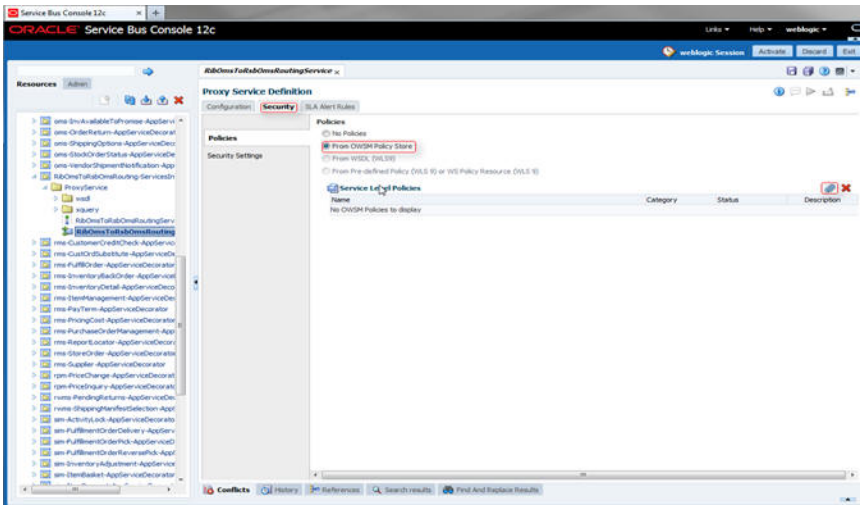
2. Click Transport Details and select the **HTTPS required** option.

Figure 2–4 HTTP Transport Configuration



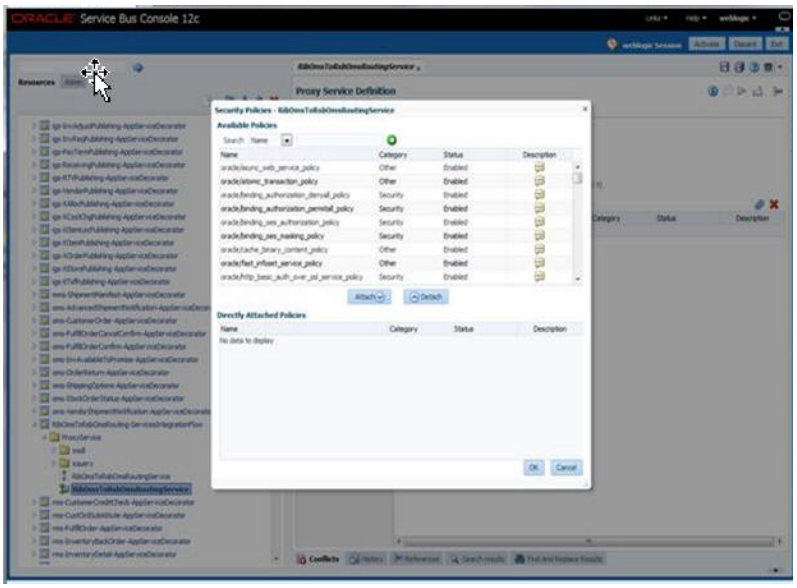
3. Click on **Security Tab** and select "From OWSM Policy Store".

Figure 2–5 Service Policy Configuration



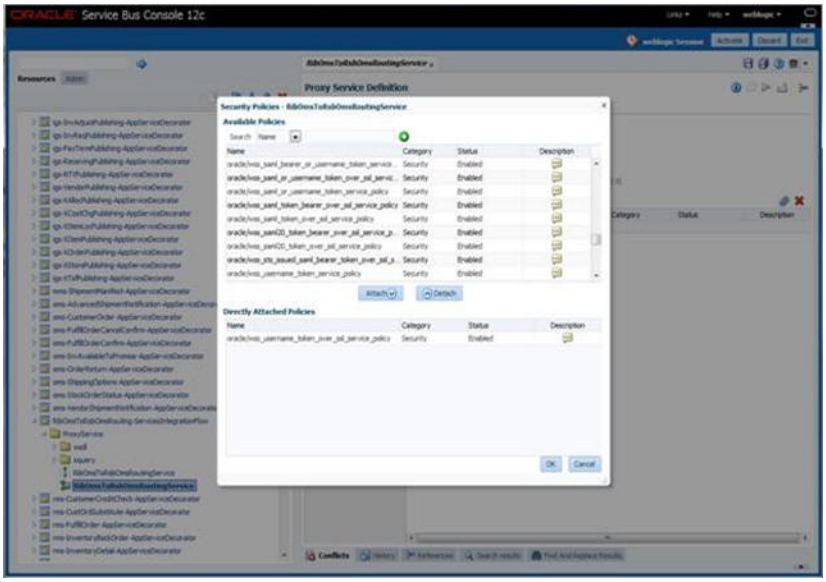
4. Click the **Attach Policies** button. The following window appears.

Figure 2–6 Attach Policies



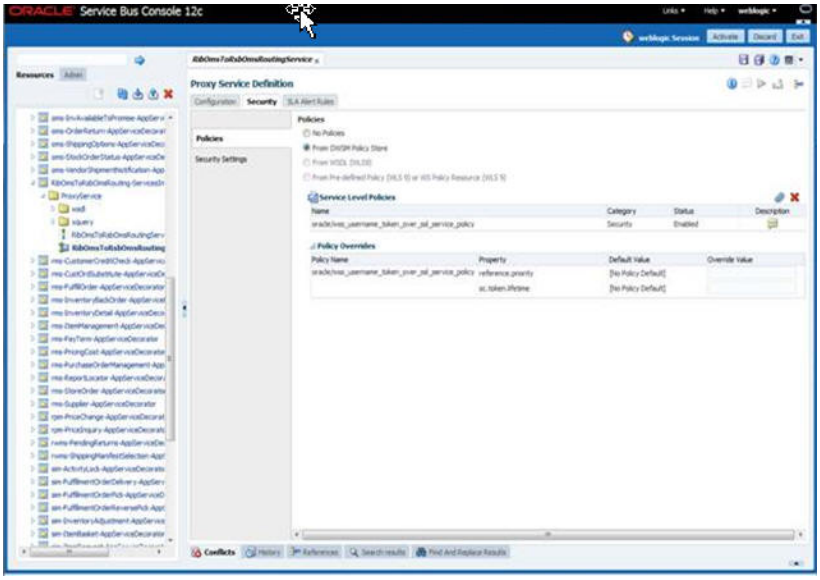
5. Select *"oracle/wss_username_token_over_ssl_service_policy"*, click **Attach** button and then click **OK** button.

Figure 2–7 Select Policy



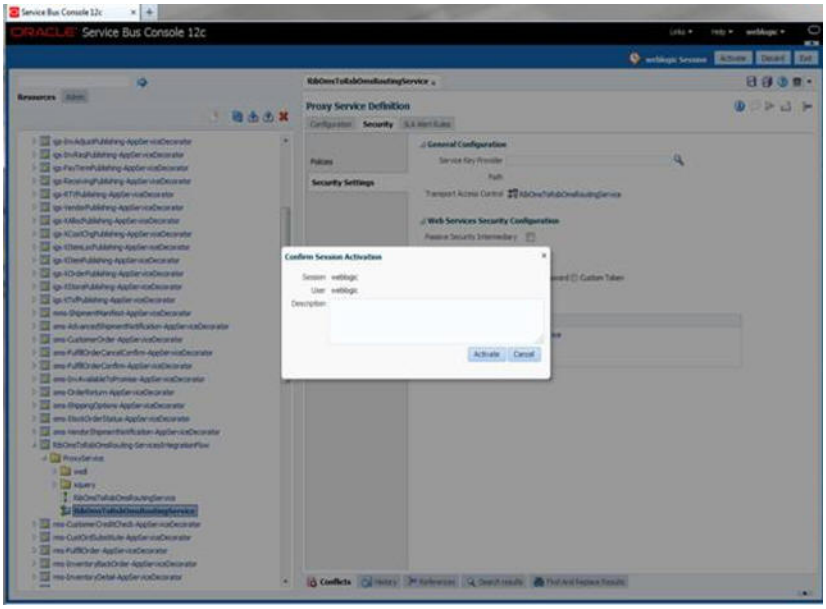
6. Click **Save** to save changes.

Figure 2–8 Save



7. Click **Activate** and **Submit** to save the session.

Figure 2–9 Activate and Save



The above steps complete securing the injector service using the OSB Console.

3

Secure IGS Web Services Using Administration Console

IGS Web services can be secured in two ways. One approach is to use a simple username and password authentication method. The second approach is to use passwords that are encrypted with certificates.

The following describes both approaches for server-side and client-side setup.

> **Note:** The various policy files that can be used to secure Web services are listed in the WS-Policy tab of the Web service in the WebLogic Server Administration Console.

Server-side Setup for Username and Password Authentication

This section describes the two-step process required for securing Web services on the server side. These steps are performed using the Oracle WebLogic Server Administration Console.

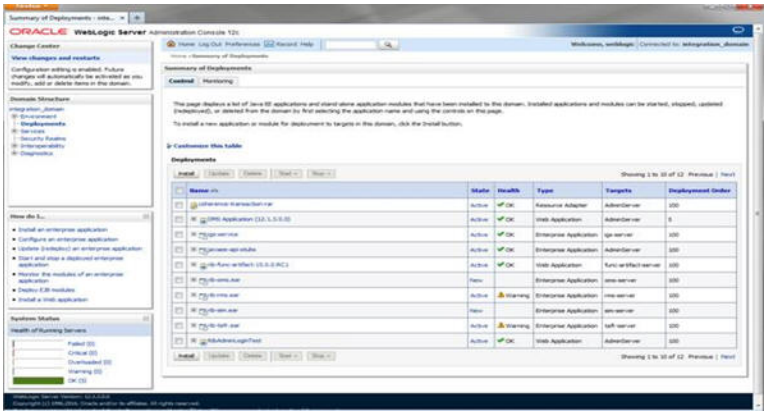
Attach the Policy File to the Web Service

The usernametoken.xml contains the policy used by the web service and is found in the META_INF/policies folder in the .ear file.

Complete the following steps to attach the policy file to a Web service:

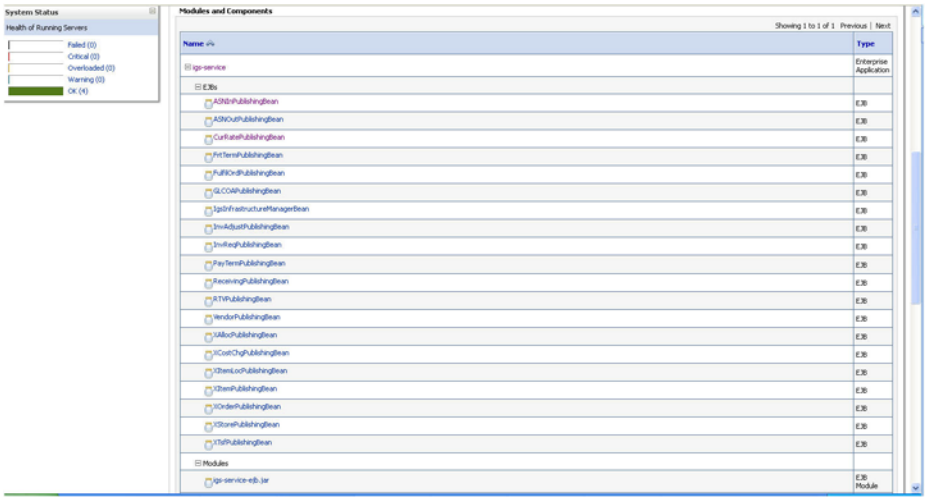
1. In the Summary of Deployments screen, click on the application. In the illustration below, the application is igs-service.

Figure 3–1 Summary of Deployments



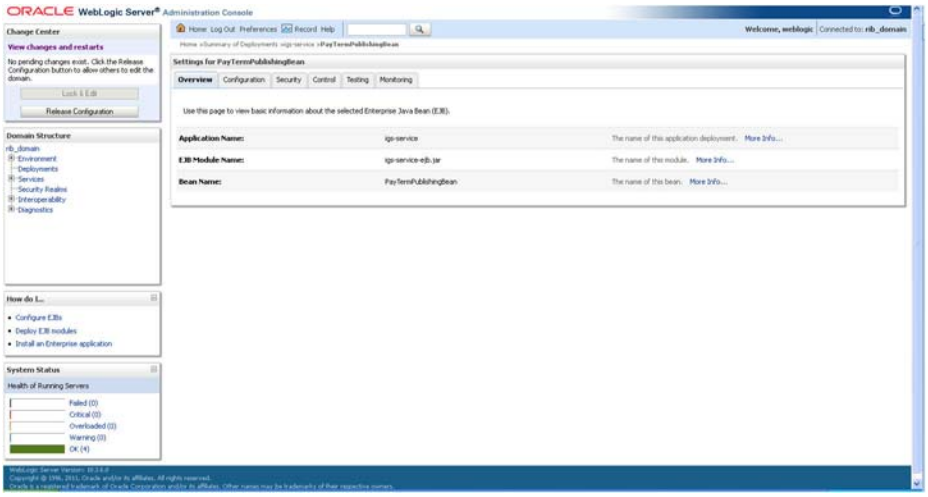
2. An overview page is displayed, including a list of modules and components installed as part of the application.

Figure 3–2 Modules and Components



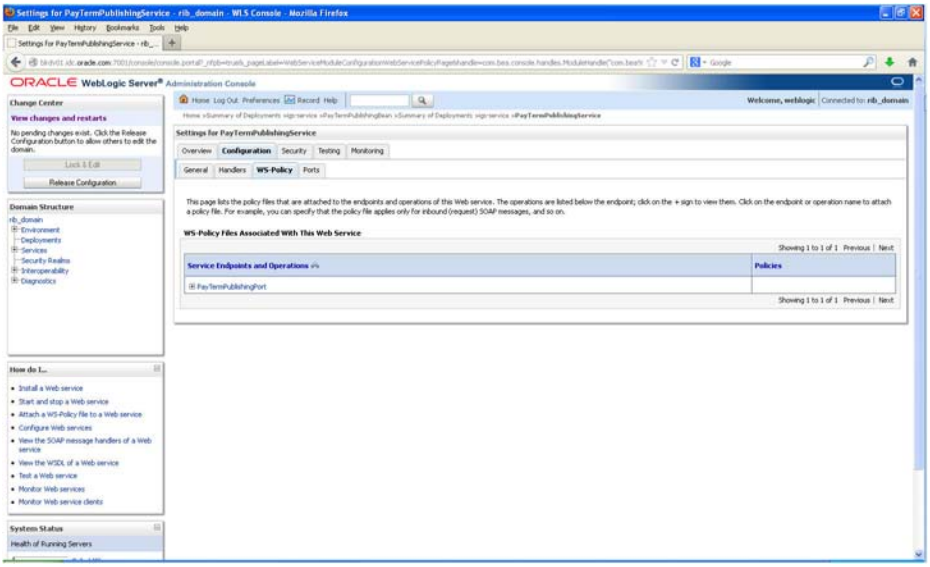
3. In the Web service list, click the service for which you want to enable security. The following screen is displayed to provide an overview of the Web service.

Figure 3–3 Settings



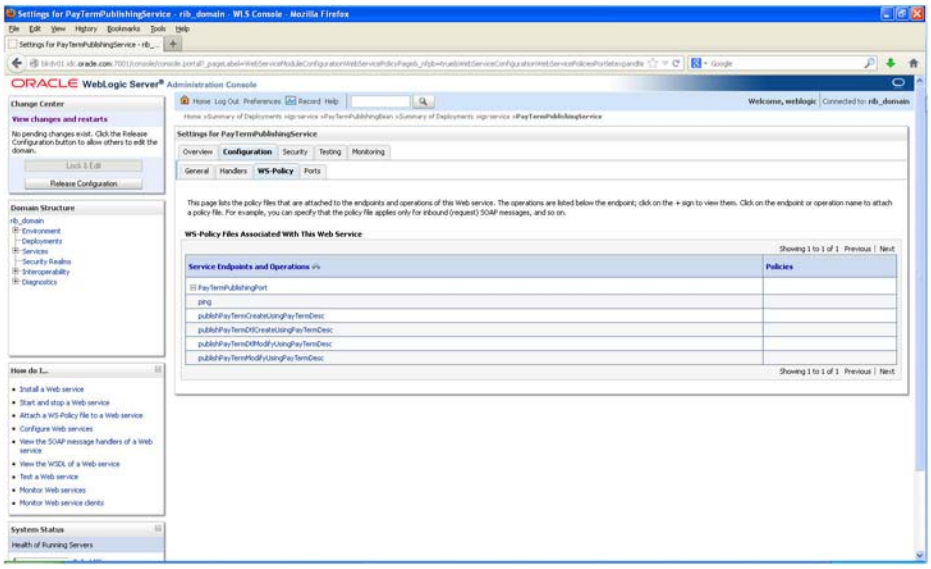
4. On this overview screen, click the **Configuration** tab. Click the **WS-Policy** tab. The Web service port is shown under **Service Endpoints and Operations**.

Figure 3–4 Settings - Configuration Tab



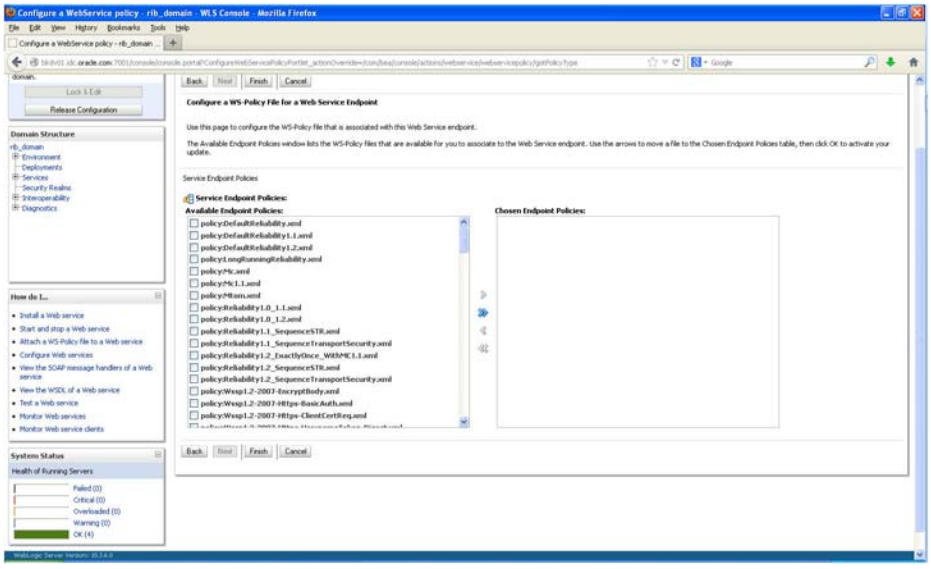
5. Click the plus sign next to the port name. The Web service operations are displayed.

Figure 3–5 Settings - Configuration Tab



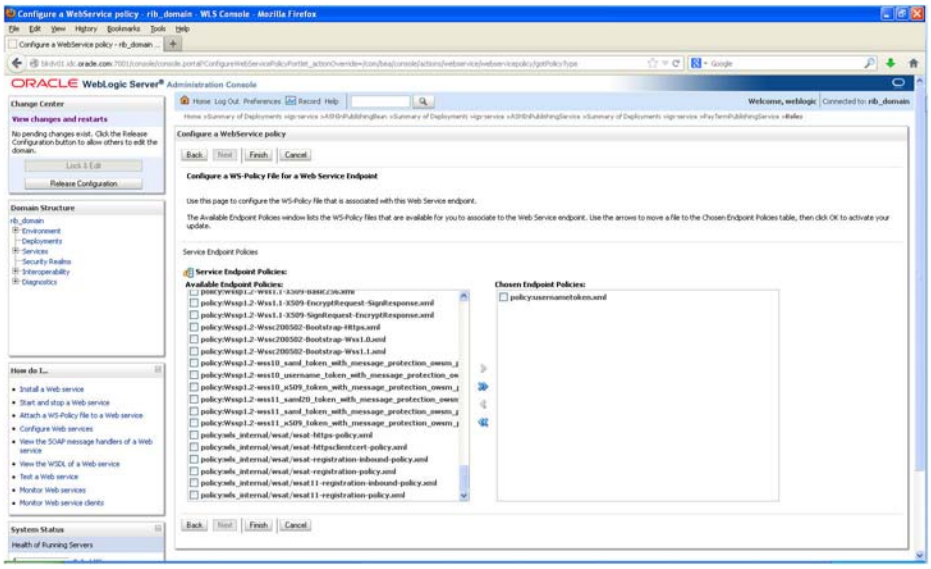
6. You can secure all the Web service operations at once or select only the operations you want to secure. Click the name of the port. On the Configure a Web Service Policy screen, you can attach the policy file to the Web service.

Figure 3–6 Configure a WS-Policy for a Web Service Endpoint



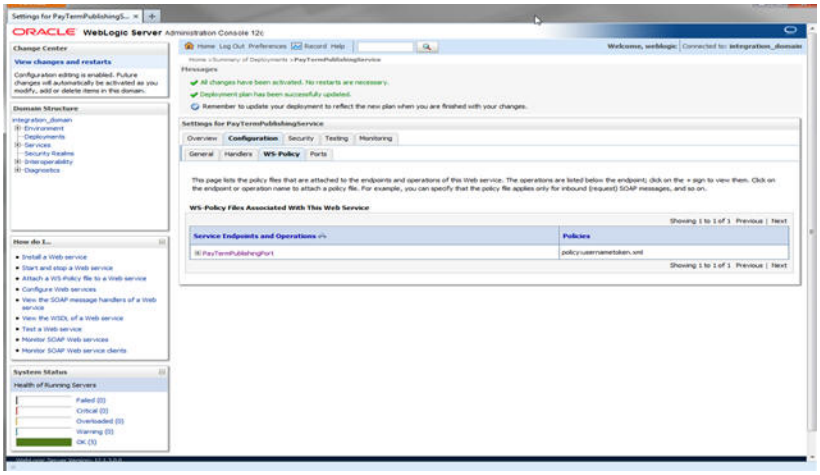
7. From the **Available Endpoint Policies** list, select policy:usernametoken.xml. Click the right arrow to move it to the drop down list below Chosen Endpoint Policies. Click **Finish**.

Figure 3–7 Configure a WS-Policy for a Web Service Endpoint



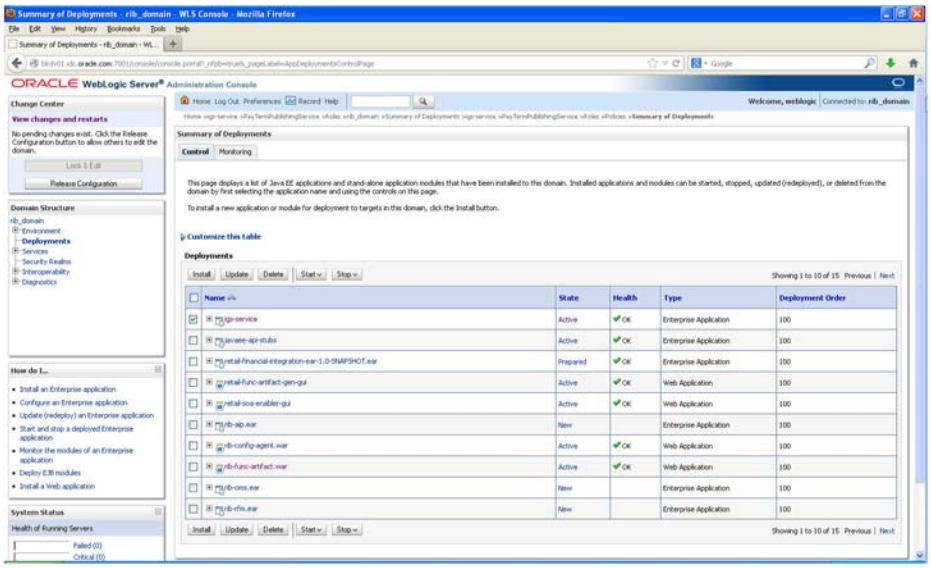
8. The following screen is displayed, including status messages near the top.

Figure 3–8 Settings - Configuration Tab



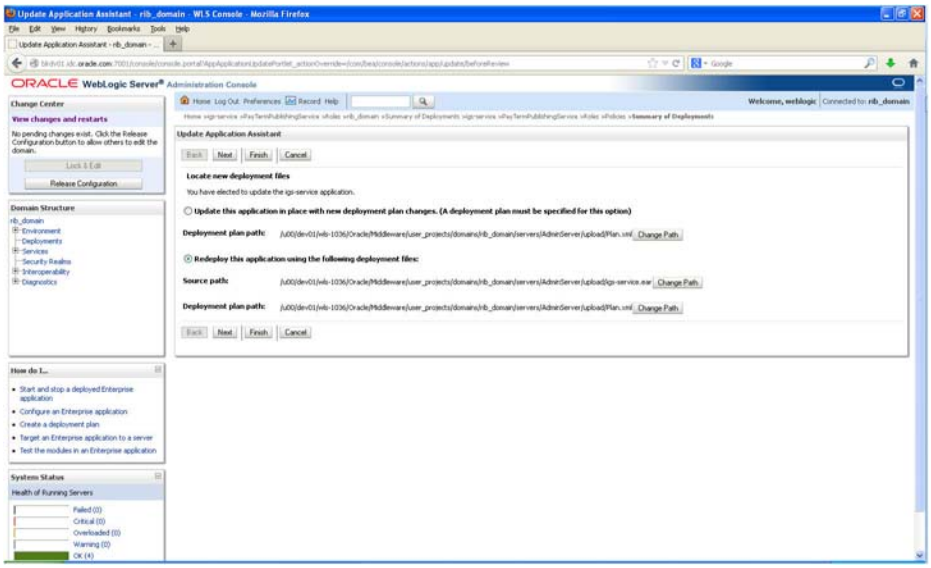
9. Now update the application to reflect the new deployment plan. Go to Deployments and select igs-service. The following screen is displayed.

Figure 3–9 Summary of Deployments



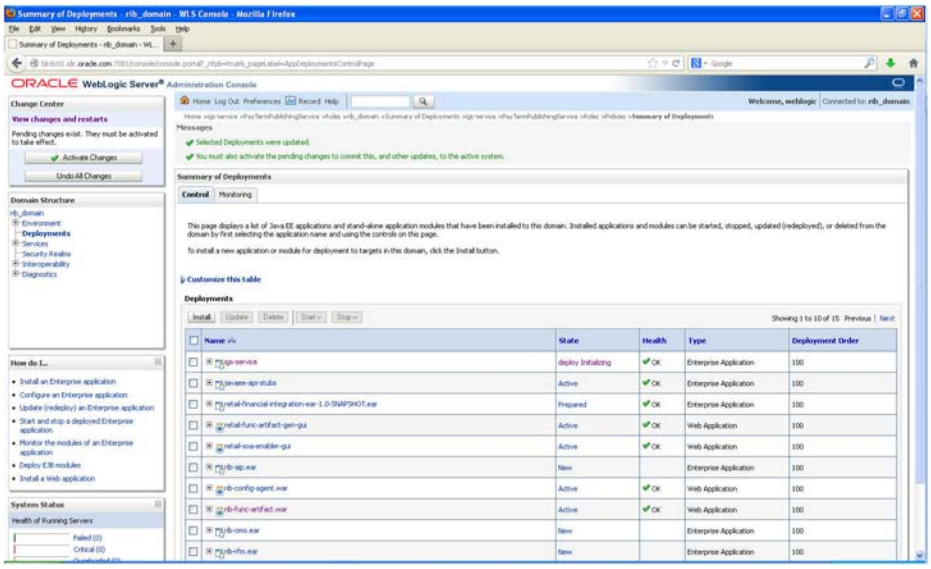
10. Click **Update** and then **Finish**.

Figure 3–10 Update Application Assistant



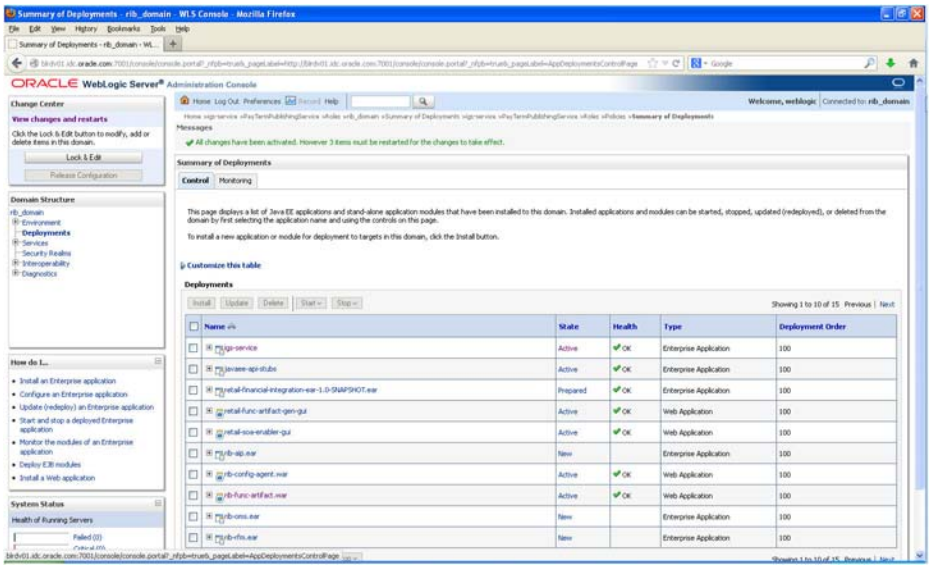
11. Click **Activate Changes**.

Figure 3–11 Summary of Deployments



12. After activating changes following screen is displayed.

Figure 3–12 Summary of Deployments



13. Under the **Testing** tab, on the Web Service page, click the WSDL to view the details of the policy just added to the Web service. The WSDL contains information similar to the following:

```
<?xml version='1.0' encoding='UTF-8'?>
<definitions
xmlns:tns="http://www.oracle.com/retail/igs/integration/services/PayTermPublish
ingService/v1"
xmlns:ns1="http://www.oracle.com/retail/integration/bus/gateway/services/Busine
ssObjectId/v1"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns2="http://www.oracle.com/retail/integration/services/exception/v1"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
```

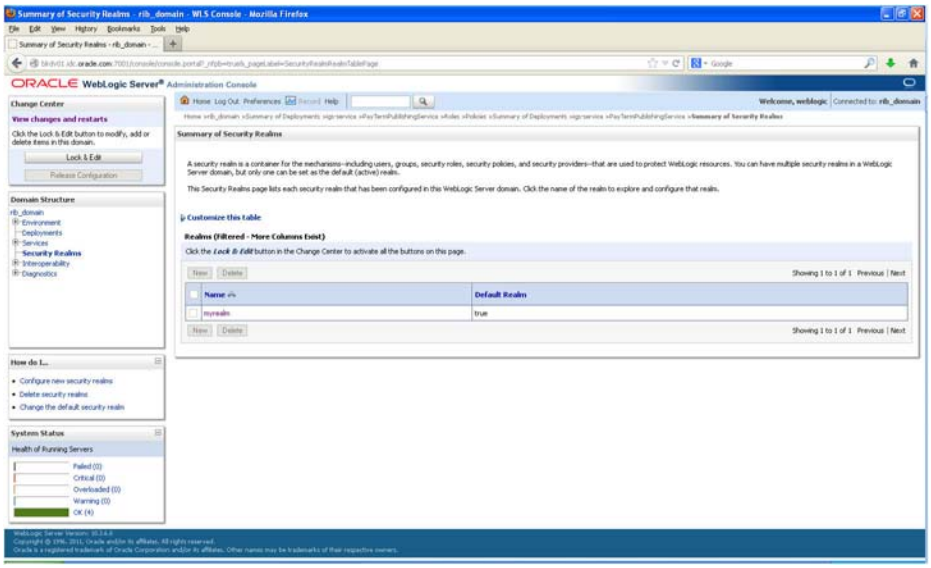
```
xmlns="http://schemas.xmlsoap.org/wsdl/" name="PayTermPublishingService"
targetNamespace="http://www.oracle.com/retail/igs/integration/services/PayTermP
ublishingService/v1"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wssutil="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecuri
ty-utility-1.0.xsd">
<wsp:UsingPolicy wssutil:Required="true" />
<wsp:Policy wssutil:Id="usernametoken">
<ns0:SupportingTokens
xmlns:ns0="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512">
<wsp:Policy>
<ns0:UsernameToken
ns0:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/Inc
ludeToken/AlwaysToRecipient">
<wsp:Policy>
<ns0:WssUsernameToken10/>
</wsp:Policy>
</ns0:UsernameToken>
</wsp:Policy>
</ns0:SupportingTokens>
</wsp:Policy>
```

Create Roles and Users

This section describes steps to add roles and users who can access the Web services. The first step is to add users to the security realm.

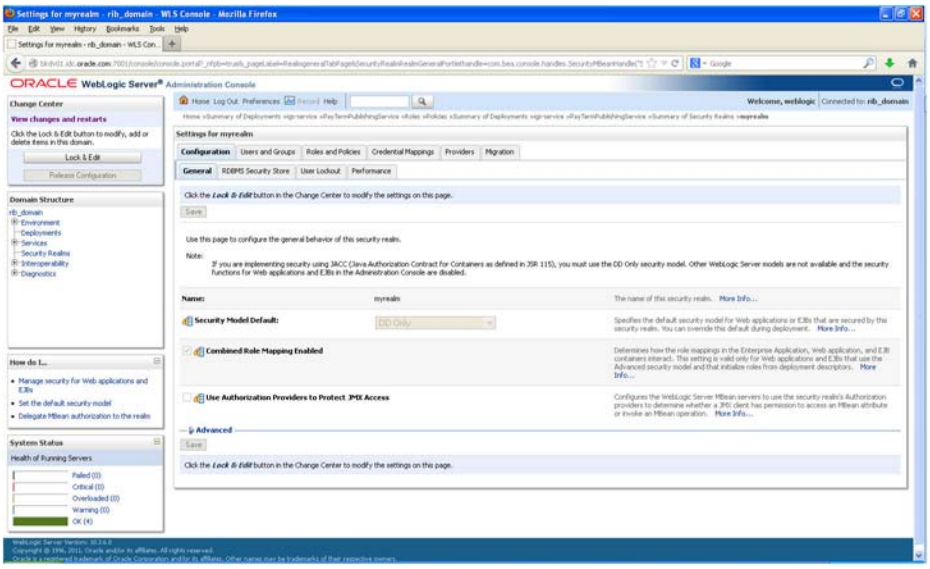
1. In the Domain Structure window of the Oracle WebLogic Services Administration Console, click the **Security Realms** link. The Summary of Security Realms screen is displayed, including the name of the default realm.

Figure 3–13 Summary of Security Realms



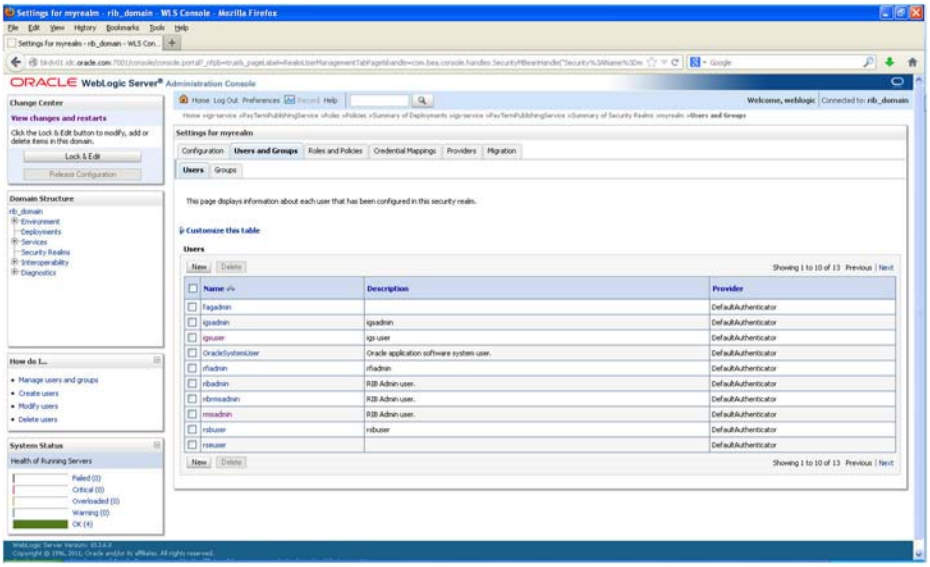
2. Click the name of the default realm. The settings for the realm are displayed.

Figure 3–14 Settings



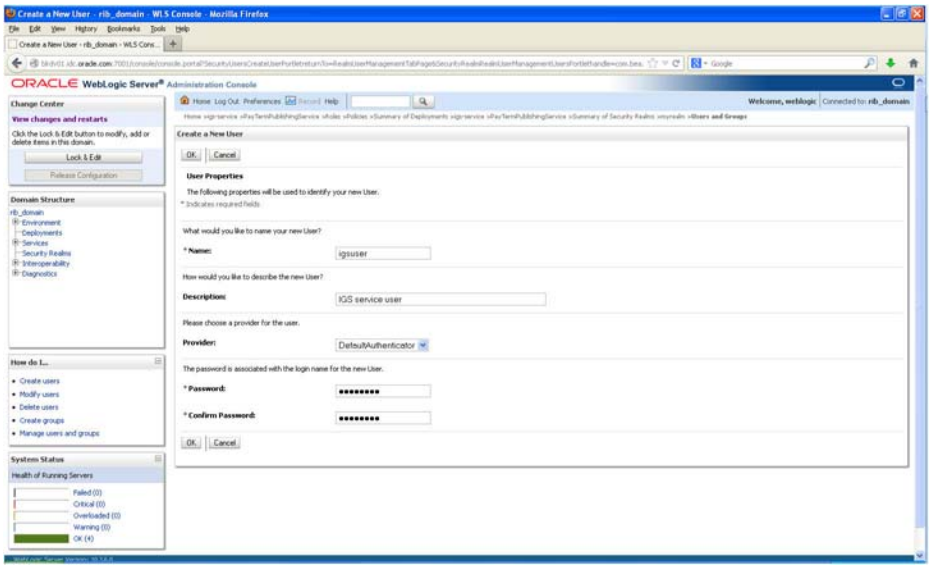
3. On the Setting screen, click the **Users and Groups** tab.

Figure 3–15 Settings



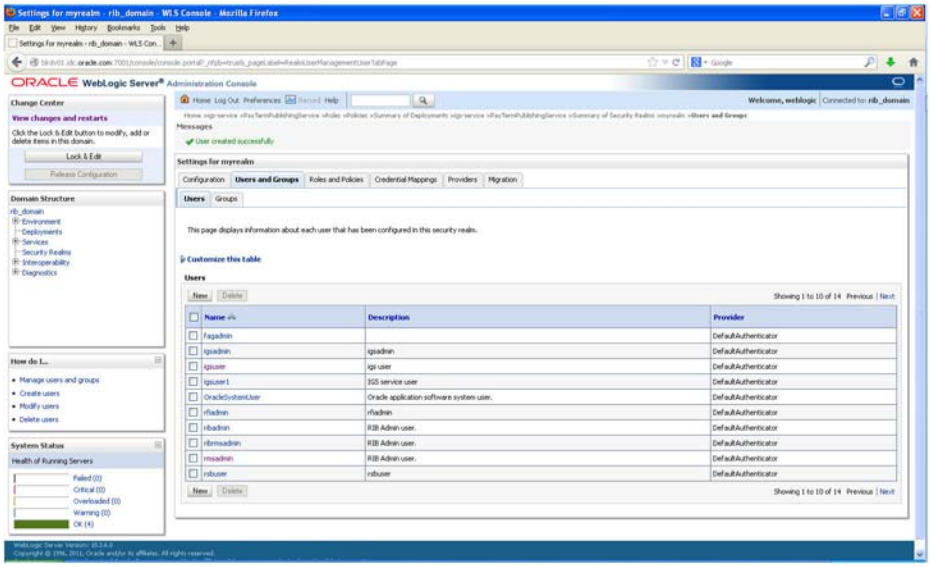
4. In the **Users and Groups** tab, click the **Users** tab. At the bottom of the Users tab, click **New**. The Create a New User screen is displayed.

Figure 3–16 Create a New User



5. In the Create a New User screen, enter a username and password. Leave the default value for Provider. Click **OK** to save the information. The new user is added to the list of users.

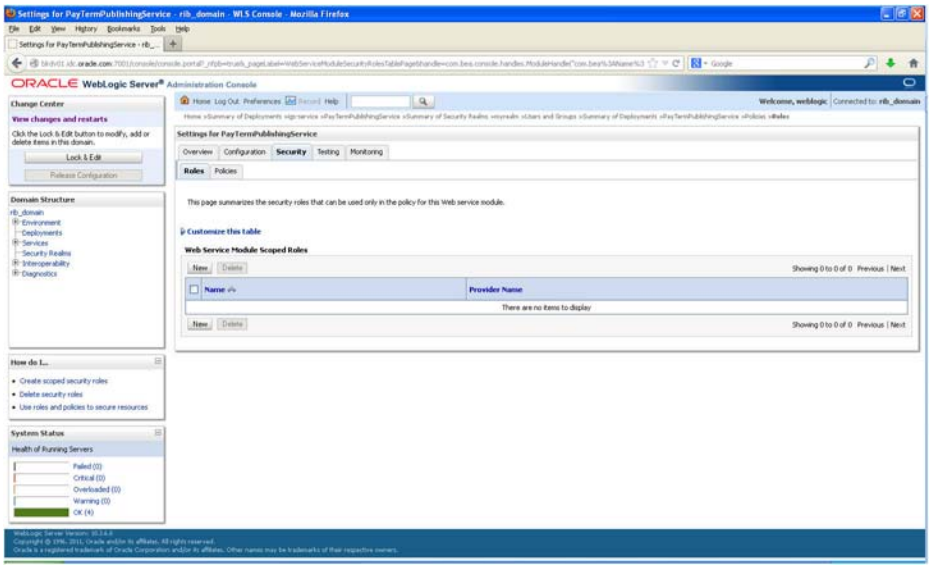
Figure 3–17 Settings



Note: You can add roles from the Roles and Policies tab of the security realm or through the Security tab of the Web service. The following instructions are for creating a role through the Security tab of the Web service.

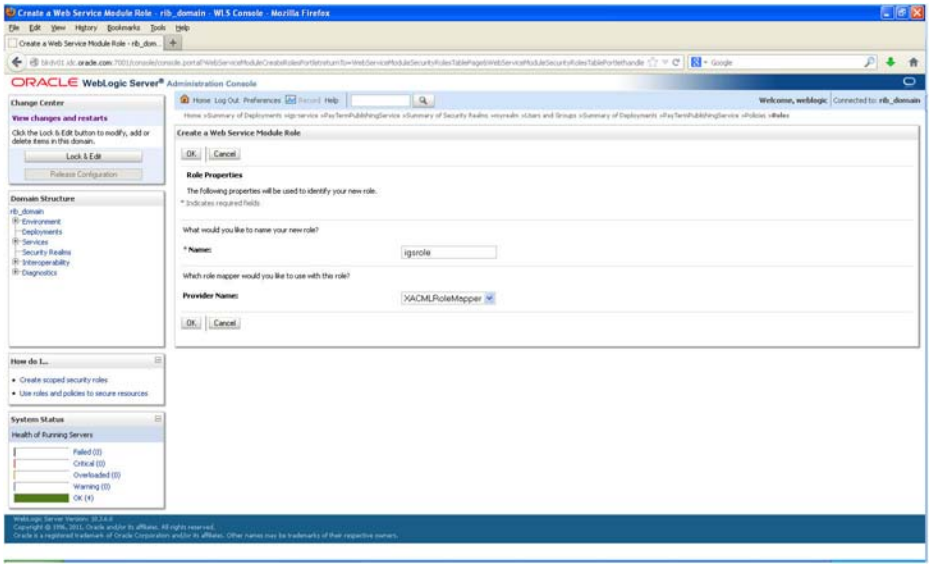
6. Navigate to the **Security** tab of the Web service. Click the **Roles** tab.

Figure 3–18 Settings for Web Service > Security > Roles Tab



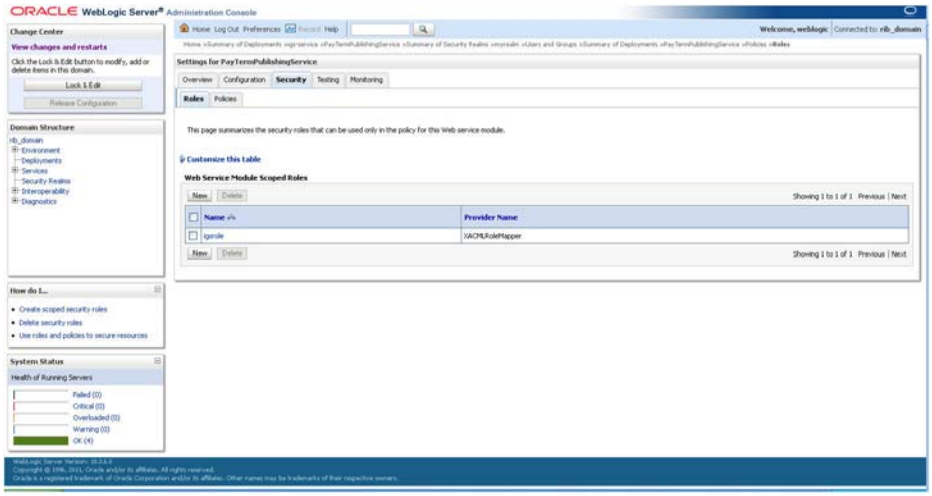
7. In the **Roles** tab, click **New**. The Create a Web Service Module Role screen is displayed.

Figure 3–19 Create a Web Service Module Role



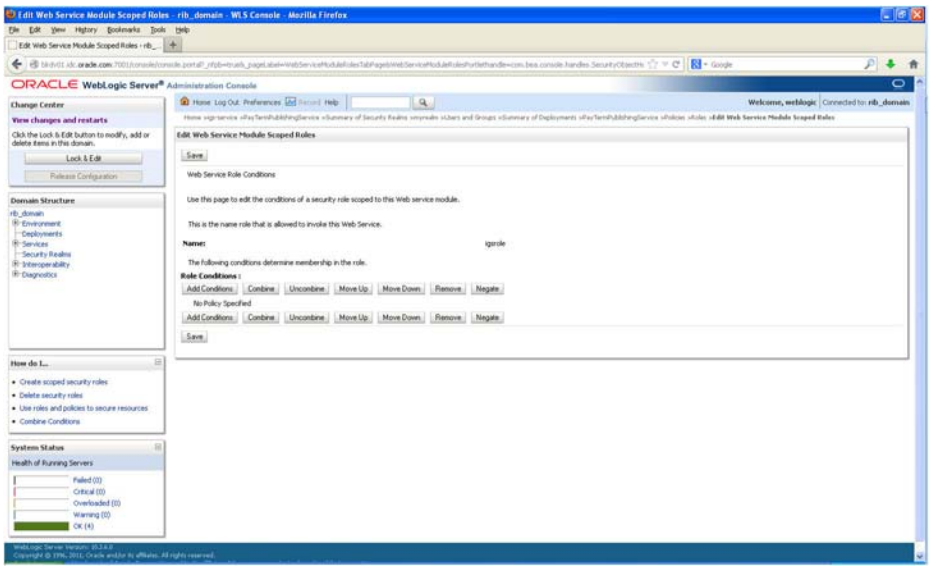
8. In the Create a Web Service Module Role screen, enter the role name in the Name field (for example, rmsrole). Leave the default value in the **Provider Name** field. Click **OK**. The new role is displayed in the **Roles** tab of the Web service.

Figure 3–20 Settings



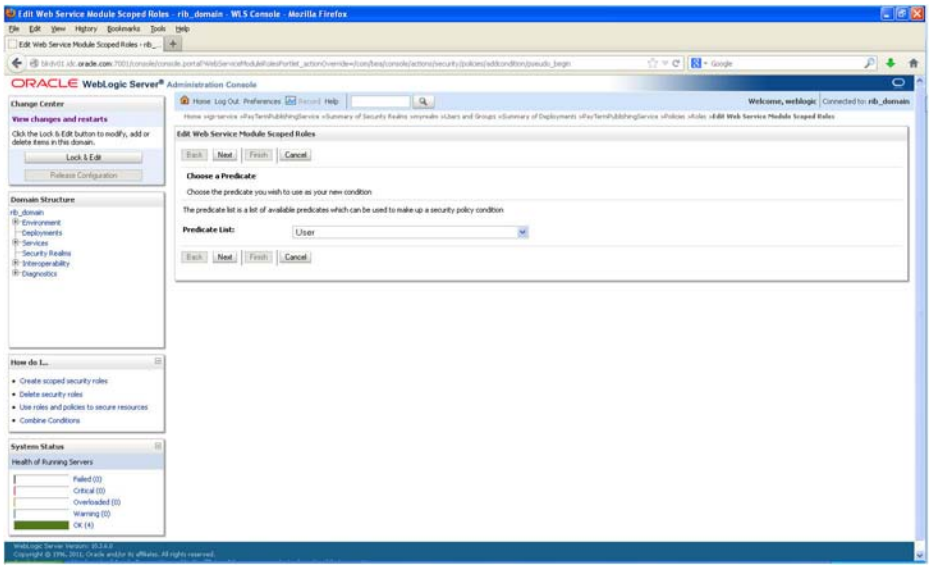
9. To add the user to the role, click the name of the new role in the **Roles** tab. The Edit Web Service Module Scoped Roles screen is displayed.

Figure 3–21 Edit Web Service Module Scoped Roles



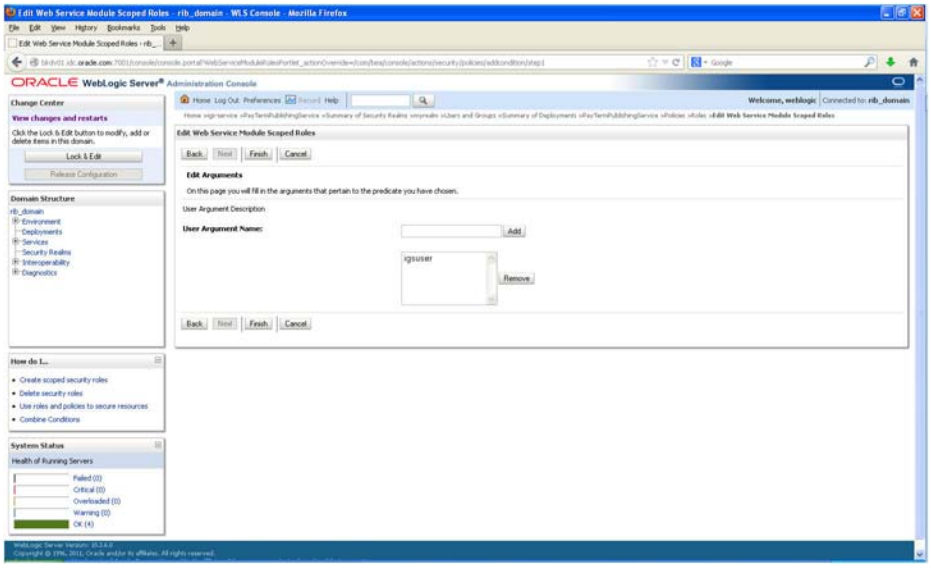
10. In the Edit Web Service Module Scoped Roles screen, click **Add Conditions**. The **Choose a Predicate** option is displayed.

Figure 3–22 Choose a Predicate



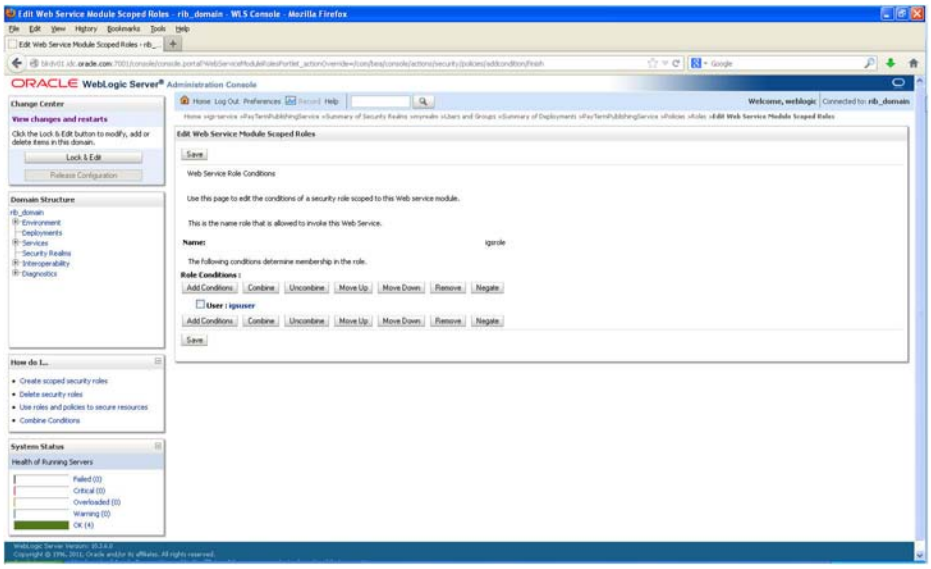
11. From the Predicate List, select **User**. Click **Next**. The **Edit Arguments** option is displayed.

Figure 3–23 Edit Web Service Module Scoped Roles



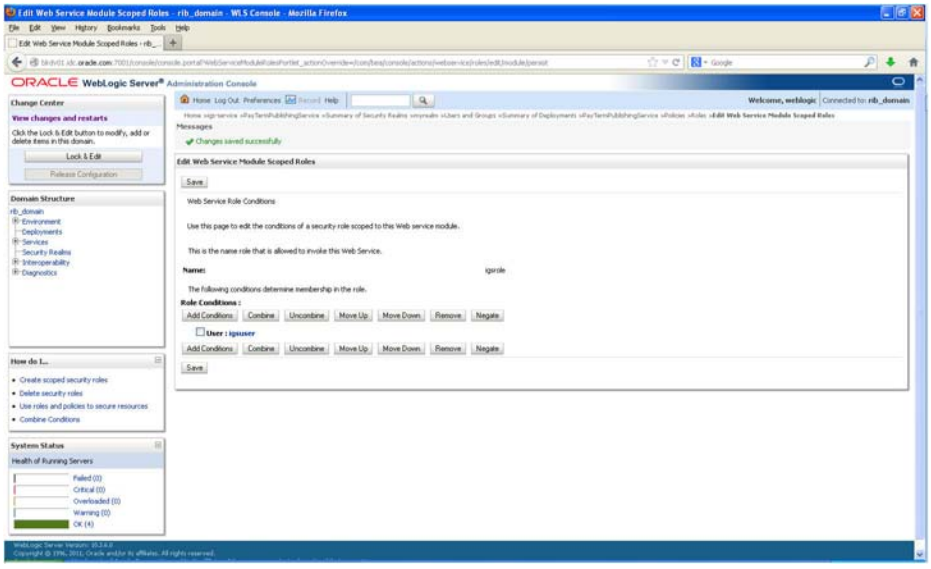
12. In the **User Argument Name** field, enter the username created in the security realm. Click **Add**. The name will move down to the box below the Add button. Click **Finish**. The following screen is displayed.

Figure 3–24 Edit Web Service Module Scoped Roles



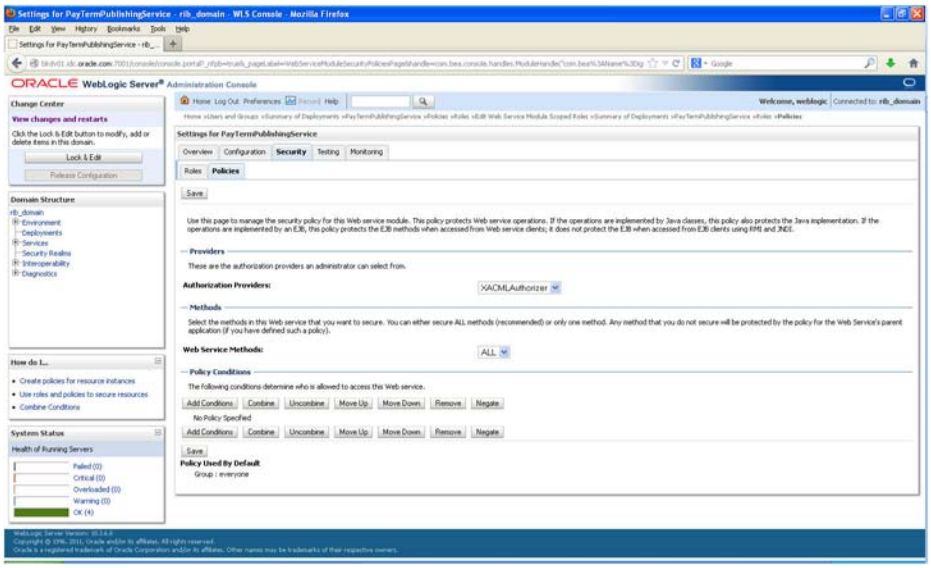
13. Click **Save**. The same screen is displayed with this message near the top: *Changes saved successfully.*

Figure 3–25 Edit Web Service Module Scoped Roles



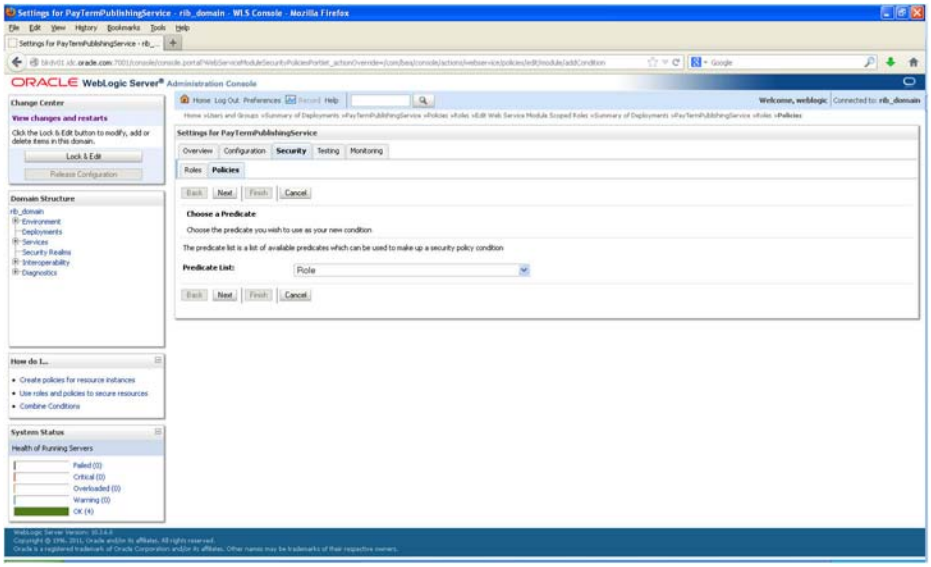
14. Return the **Security** tab of the Web service and click the **Policies** tab.

Figure 3–26 Settings



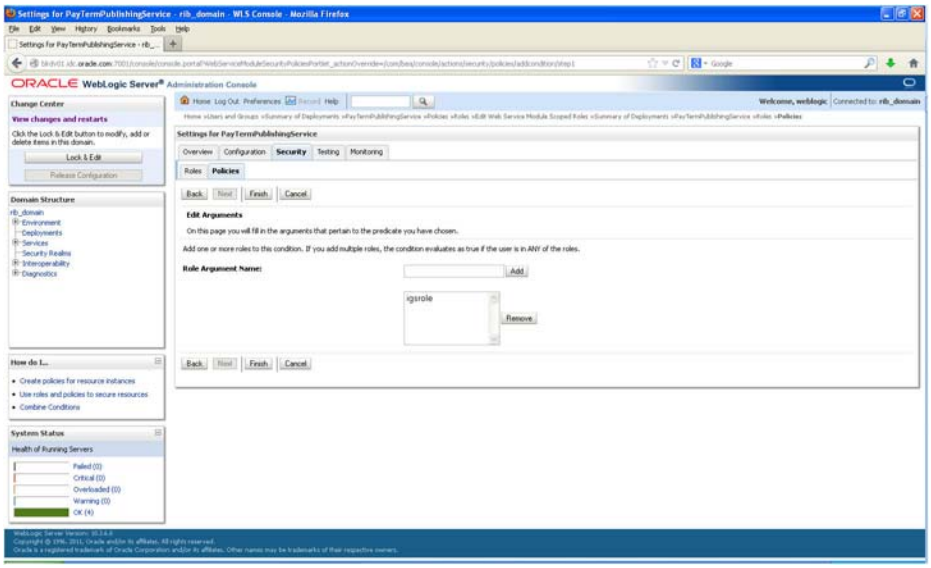
15. On the **Policies** tab, click **Add Conditions**. The **Choose a Predicate** option is displayed.

Figure 3–27 Settings



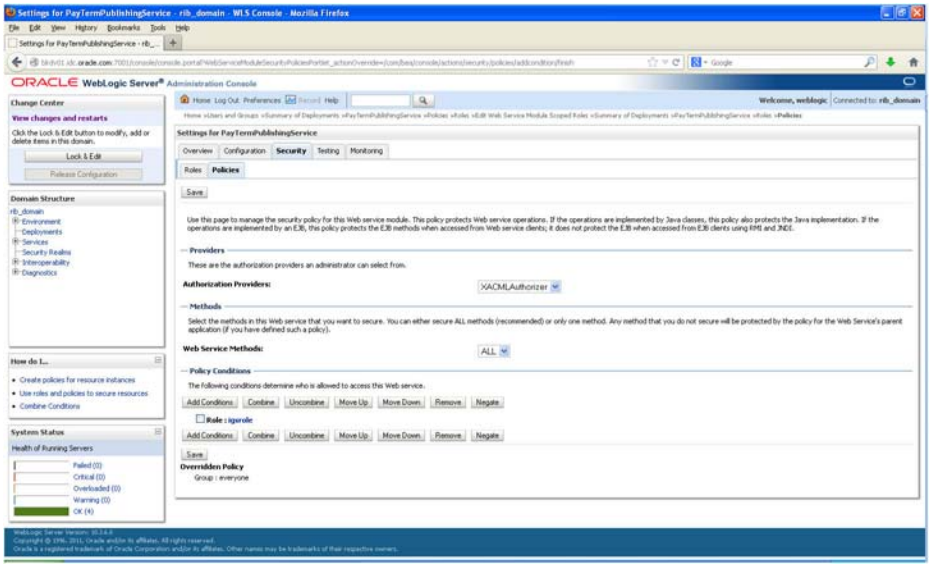
16. From the Predicate List, select **Role**. Click **Next**. The **Edit Arguments** option is displayed.

Figure 3–28 Settings



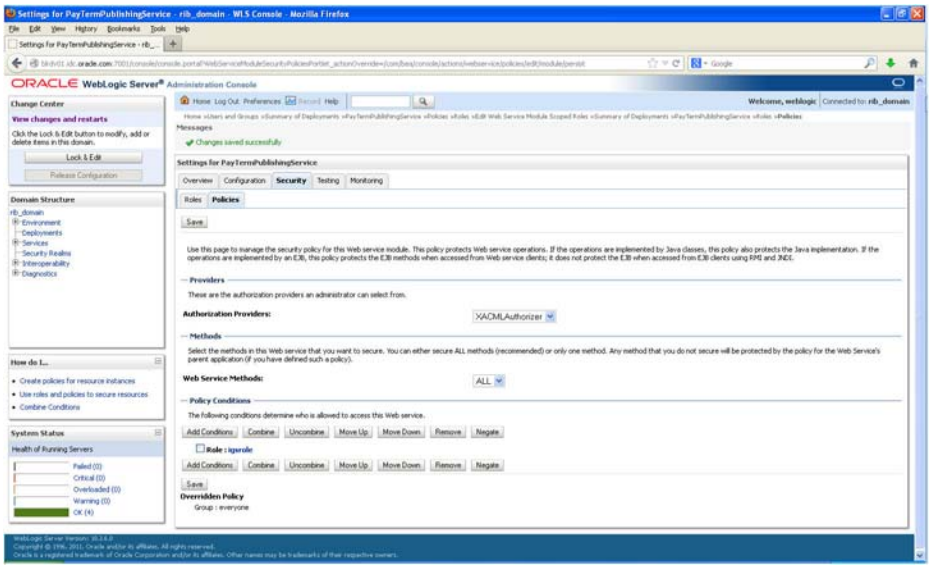
17. In the **Role Argument Name** field, enter the role name created earlier. Click **Add**. The role name will move down to the box below the **Add** button. Click **Finish** to return to the **Policy Conditions** screen.

Figure 3–29 Settings



18. Click **Save**. The Policy Conditions screen is displayed with this message near the top: *Changes saved successfully*.

Figure 3–30 Settings



Client-side Setup for Username and Password Authentication

The following is sample code for calling a secure IGS Web service.

> **Note:** The following is sample code for invoking the *PayTermPublishingService* service. When you generate Java consumer for a Web service, the generated jar file contains classes specific to that Web service. Use the appropriate classes in the client code. Service namespace and WSDL location also should be changed accordingly.

```
package com.oracle.retail.rms.client;

import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import javax.xml.namespace.QName;
import javax.xml.ws.BindingProvider;
import
com.oracle.retail.igs.integration.services.paytermpublishingservice.v1.PayTermPubl
ishingPortType;
import
com.oracle.retail.igs.integration.services.paytermpublishingservice.v1.PayTermPubl
ishingService;
import
com.oracle.retail.igs.integration.services.paytermpublishingservice.v1.PublishPayT
ermCreateUsingPayTermDesc;
import
com.oracle.retail.igs.integration.services.paytermpublishingservice.v1.PublishPayT
ermCreateUsingPayTermDescResponse;
import com.oracle.retail.integration.base.bo.paytermdesc.v1.PayTermDesc;
import weblogic.wsee.security.unt.ClientUNTCredentialProvider;
import weblogic.xml.crypto.wss.WSSecurityContext;
import weblogic.xml.crypto.wss.provider.CredentialProvider;
import junit.framework.TestCase;
```

```
public class PayTermPublishingClient extends TestCase {
public void testCreatePayTermDesc() {
try {

// qName is namespace of the service
QName qName = new
QName("http://www.oracle.com/retail/igs/integration/services/PayTermPublishingServ
ice/v1"," PayTermPublishingService");

// url is the URL of the WSDL of the web service
URL url = new
URL("http://igshost.example.com:18030/PayTermPublishingBean/PayTermPublishingServi
ce?WSDL");

// Create an instance of the web service
PayTermPublishingServiceservice = new PayTermPublishingService (url,qName);
PayTermPublishingPortType =   service.getPayTermPublishingPort ();

// Set the security credentials in the service context
List credProviders = new ArrayList();
CredentialProvider cp = new ClientUNTCredentialProvider("<rms user>","<rms
password>");
credProviders.add(cp);
Map<String, Object> rc =            ((BindingProvider)port).getRequestContext();
rc.put(WSSecurityContext.CREDENTIAL_PROVIDER_LIST, credProviders);

// Populate the service method input object
PayTermDesc payTermDesc = new PayTermDesc();
payTermDesc.setTerms("terms");
PublishPayTermCreateUsingPayTermDesc payTermCreateDesc = new P
PublishPayTermCreateUsingPayTermDesc();
payTermCreateDesc.setPayTermDesc(payTermDesc);

// Call the web service
PublishPayTermCreateUsingPayTermDescResponse response =
port.publishPayTermCreateUsingPayTermDesc(payTermCreateDesc,"1");
System.out.println("response="+response);
}catch(Exception e){
  e.printStackTrace();
}
}
}
```

Server-side Setup for Encrypted Username and Password Token Authentication

WebLogic provides predefined policy files for securing Web services. This section describes the process required to secure a Web service where username and password are encrypted and signed.

Take the following steps to secure the Web service:

1. Follow the steps to attach the policy file to the Web service described in the section, *Attach Policy File to the Web Service*, with this exception: In Step 7, select *policy*:*Wssp1.2-2007-Wss1.1-UsernameToken-Plain-X509-Basic256.xml* (instead of *policy:usernametoken.xml*). Follow the remaining steps as described.

   After attaching the policy file, the header for the WSDL of the Web service contains the following:

```
<wsp:UsingPolicy wssutil:Required="true"/>
<wsp:Policy
wssutil:Id="Wssp1.2-2007-Wss1.0-UsernameToken-Plain-X509-Basic256.xml">
<ns1:AsymmetricBinding
xmlns:ns1="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
<wsp:Policy>
<ns1:InitiatorToken>
<wsp:Policy>
<ns1:X509Token
ns1:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/Inc
ludeToken/AlwaysToRecipient">
<wsp:Policy>
<ns1:WssX509V3Token10/>
</wsp:Policy>
</ns1:X509Token>
</wsp:Policy>
</ns1:InitiatorToken>
<ns1:RecipientToken>
<wsp:Policy>
<ns1:X509Token
ns1:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/Inc
ludeToken/Never">
<wsp:Policy>
<ns1:WssX509V3Token10/>
</wsp:Policy>
</ns1:X509Token>
</wsp:Policy>
</ns1:RecipientToken>
<ns1:AlgorithmSuite>
<wsp:Policy>
<ns1:Basic256/>
</wsp:Policy>
</ns1:AlgorithmSuite>
<ns1:Layout>
<wsp:Policy>
<ns1:Lax/>
</wsp:Policy>
</ns1:Layout>
<ns1:IncludeTimestamp/>
<ns1:ProtectTokens/>
<ns1:OnlySignEntireHeadersAndBody/>
</wsp:Policy>
</ns1:AsymmetricBinding>
<ns2:SignedEncryptedSupportingTokens
xmlns:ns2="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
<wsp:Policy>
<ns2:UsernameToken
ns2:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/Inc
ludeToken/AlwaysToRecipient">
<wsp:Policy>
<ns2:WssUsernameToken10/>
</wsp:Policy>
</ns2:UsernameToken>
</wsp:Policy>
</ns2:SignedEncryptedSupportingTokens>
<ns3:Wss10
xmlns:ns3="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
<wsp:Policy>
<ns3:MustSupportRefKeyIdentifier/>
<ns3:MustSupportRefIssuerSerial/>
```

```
</wsp:Policy>
</ns3:Wss10>
</wsp:Policy>
```

2. The key combination used by the client to sign the message is a valid one for the server. The client certificate must be signed with a certificate authority that is trusted by the server.
3. WebLogic instances include a demo CA. The certificate and key for it is in *$WL_HOME/Middleware/wlserver/server/lib/CertGenCA.der* and *CertGenCAKey.der*. The key does not appear to change between WebLogic installations and is trusted by the default *DemoTrust* store. For this reason, the DemoTrust store must never be enabled in a production environment. Otherwise anybody can become "trusted" fairly easily.
4. WebLogic CertGen command can be used for generating keys of the correct key length and signing them with the demo CA noted above. A client certification/key pair is required to sign the outgoing message and server certificate to encrypt the critical information.

   ```
   java -classpath $WL_HOME/Middleware/wlserver/server/lib/weblogic.jar
   utils.CertGen -certfile ClientCert -keyfile ClientKey -keyfilepass ClientKey
   -cn <rms user>
   ```

   The above command generates the following files:

   1. ClientCert.der
   2. ClientCert.pem
   3. ClientKey.der
   4. ClientKey.pem

   In the above example of a command, the username is <rms user>. Replace <rms user> with the username of the user who will access the Web service.
5. The command below generates the four files that follow it:

   ```
   java -classpath $WL_HOME/Middleware/wlserver/server/lib/weblogic.jar
   utils.CertGen -certfile ServerCert -keyfile ServerKey -keyfilepass ServerKey
   -cn <rms user>
   ```

   1. ServerCert.der
   2. ServerCert.pem
   3. ServerKey.der
   4. ServerKey.pem

   In the above example of a command, the username is <rms user>. Replace <rms user> with the username of the user who will access the Web service.
6. Using the following commands, import the files into key stores:

   ```
   java -classpath $WL_HOME/Middleware/wlserver/server/lib/weblogic.jar
   utils.ImportPrivateKey -certfile ClientCert.der -keyfile ClientKey.der
   -keyfilepass ClientKey -keystore ClientIdentity.jks -storepass ClientKey -alias
   identity - keypass ClientKey

   java -classpath $WL_HOME/Middleware/wlserver/server/lib/weblogic.jar
   utils.ImportPrivateKey -certfile ServerCert.der -keyfile ServerKey.der
   -keyfilepass ServerKey -keystore ServerIdentity.jks -storepass ServerKey -alias
   identity - keypass ServerKey
   ```

7. Using the script in Appendix: configWss.py, configure the WebLogic server to use the key. Copy the script and save it in the location from which it will run.

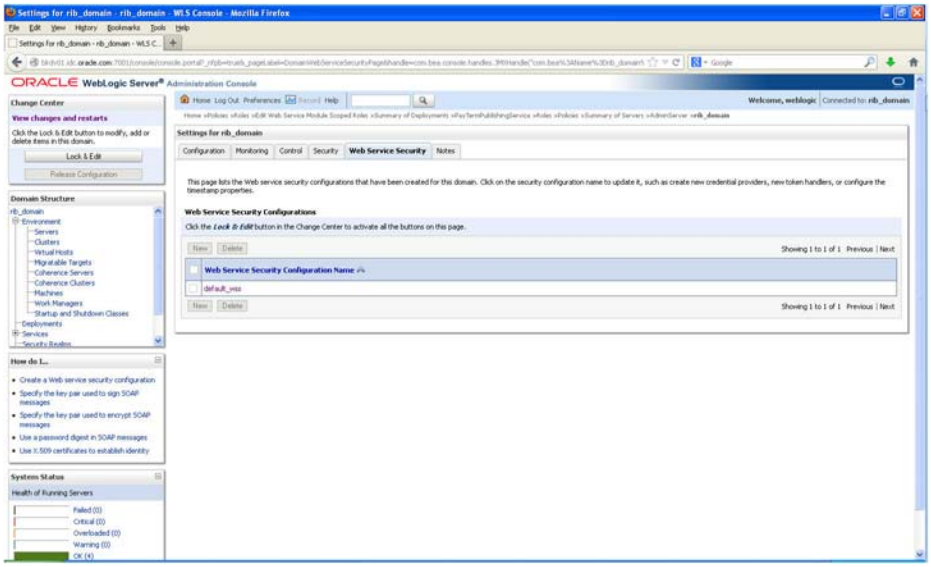
   ```
   java -classpath $WL_HOME/Middleware/wlserver/server/lib/weblogic.jar
   weblogic.WLST configWss.py <weblogicuser> <WebLogic Password> <weblogichost>
   <weblogic admin port> ServerIdentity.jks ServerKey identity ServerKey
   ```

   For example:

   ```
   java -classpath $WL_HOME/Middleware/wlserver/server/lib/weblogic.jar
   weblogic.WLST configWss.py weblogic <WebLogic Password> localhost
   7001/home/wls/ServerIdentity.jks ServerKey identity ServerKey
   ```

8. In the WebLogic logic console, check the **Web Service Security** tab to verify that the command ran properly. Note that the default_ww configuration is used for all Web services unless otherwise indicated.

Figure 3–31 Settings



9. After the certificate setup is completed for the Web service, follow the steps in the "Create Roles and Users" section to create a user in WebLogic to access the Web service.

10. Restart the server. Create a client to invoke the Web service.

Client-side Setup for Encrypted Username and Password Token Authentication

The following is sample code for calling a Web service that is secured using the policy file, policy:Wssp1.2-2007-Wss1.1-UsernameToken-Plain-X509-Basic256.xml:

```
package com.test;
import java.net.URL;
import java.security.cert.X509Certificate;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import javax.xml.namespace.QName;
import javax.xml.ws.BindingProvider;
```

```
import javax.xml.ws.WebServiceRef;
import
com.oracle.retail.igs.integration.services.paytermpublishingservice.v1.PayTermPubl
ishingPortType;
import
com.oracle.retail.igs.integration.services.paytermpublishingservice.v1.PayTermPubl
ishingService;
import
com.oracle.retail.igs.integration.services.paytermpublishingservice.v1.PublishPayT
ermCreateUsingPayTermDesc;
import
com.oracle.retail.igs.integration.services.paytermpublishingservice.v1.PublishPayT
ermCreateUsingPayTermDescResponse;
import com.oracle.retail.integration.base.bo.paytermdesc.v1.PayTermDesc;
import weblogic.security.SSL.TrustManager;
import weblogic.wsee.security.bst.ClientBSTCredentialProvider;
import weblogic.wsee.security.unt.ClientUNTCredentialProvider;
import weblogic.wsee.security.util.CertUtils;
import weblogic.xml.crypto.wss.WSSecurityContext;
import weblogic.xml.crypto.wss.provider.CredentialProvider;
public class Client {
public static void main(String args[]){
try {
//qName is namespace of the service
QName qName = new
QName("http://www.oracle.com/retail/igs/integration/services/PayTermPublishingServ
ice/v1"," PayTermPublishingService");

// url is the URL of the WSDL of the web service
URL url = new
URL("http://igshost.example.com:18030/PayTermPublishingBean/PayTermPublishingServi
ce?WSDL";

// Create an instance of the web service
PayTermPublishingServiceservice = new PayTermPublishingService(url,qName);
PayTermPublishingPortType = service.getPayTermPublishingPort ();
PayTermDesc payTermDesc = new PayTermDesc();
payTermDesc.setTerms("terms");
PublishPayTermCreateUsingPayTermDesc payTermCreateDesc = new
PublishPayTermCreateUsingPayTermDesc();
payTermCreateDesc.setPayTermDesc(payTermDesc);

String serverCertFile = "D:/head/retail-soa-enabler/dist/client/ServerCert.der";
String clientKeyStore =
"D:/head/retail-soa-enabler/dist/client/ClientIdentity.jks";
String clientKeyStorePass = "ClientKey";
String clientKeyAlias = "identity";
String clientKeyPass = "ClientKey";

List credProviders = new ArrayList();
ClientUNTCredentialProvider unt =  new ClientUNTCredentialProvider("<rms
user>","<rms password>");
credProviders.add(unt);
final X509Certificate serverCert =
(X509Certificate)CertUtils.getCertificate(serverCertFile);
serverCert.checkValidity();

CredentialProvider cp = new ClientBSTCredentialProvider(clientKeyStore,
clientKeyStorePass,clientKeyAlias, clientKeyPass, "JKS", serverCert);
credProviders.add(cp);
```

```
Map requestContext = ((BindingProvider)port).getRequestContext();
requestContext.put(WSSecurityContext.CREDENTIAL_PROVIDER_LIST, credProviders);
requestContext.put(WSSecurityContext.TRUST_MANAGER,new TrustManager() {
public boolean certificateCallback(X509Certificate[] chain,int validateErr) {
boolean result = chain[0].equals(serverCert);
return result;
}
});

PublishPayTermCreateUsingPayTermDescResponse response =
port.publishPayTermCreateUsingPayTermDesc(payTermCreateDesc,"1");
System.out.println("response="+response);

} catch(Exception e){
e.printStackTrace();
}
}
}
```

4

Security Feature Overview

Caution: Oracle is not responsible for the security compliance of any product customization performed by a retailer, system integrator, or reseller.

The relevant security features fall into one or more of the following categories. For information on these categories, see the following sections:

- Securing Sensitive Data
- Securing the Application
- Securing the Application Environment and Configuration

Securing Sensitive Data

The protection of sensitive data during transit, processing, and storage is paramount. Sensitive data includes personally identifiable information such as credit card number, Social Security number, checking account number, and positive ID such as driver's license number. The Oracle Retail Integration Bus focuses on protecting sensitive data.

Cardholder Data

RIB, being an integration application, does not store credit card data. The applications getting integrated through RIB handle all access to cardholder data and supply tokens to use in place of actual cardholder data.

Communication with web service Application

An additional layer of communication security is provided for web application (example: OMS). These applications require the use of a Secure Socket Layer (SSL) to access them.SSL provides an additional layer of encryption and security of the information sent to and received from these applications.

Securing the Application

Securing access to the application against malicious attacks and auditing secure events are accomplished with passwords, additional testing of Web applications, and additional examination of source code.

Passwords

The RIB administration user interface username and password for accessing the user interface are created and stored inside the WebLogic Server security realm, and are protected by your WebLogic security configuration. For more information, see the *Oracle Retail Integration Bus Installation Guide*.

Default Accounts and Passwords

RIB applications do not contain any default accounts, user IDs, or passwords. An application username and password are entered by the user during the installation process.

Tools

RIB uses the Fortify 360 tool to scan for security issues. As with any tool, the output of this tool should be analyzed in detail since the output may contain false positive warnings. You can use any tools that you choose. No recommendation of the following tool is intended or implied. Fortify 360 is a tool that analyzes software for vulnerabilities. The static analysis component examines an application's source code for potentially exploitable vulnerabilities. The dynamic analysis component identifies vulnerabilities that can be found only when an application is running. All vulnerabilities can be ranked according to their relevance. Fortify can be found at the following website: `http://www.fortify.com`

Securing the Application Environment and Configuration

Securing the application environment and configuration covers the following areas:

- Database

Database

If sensitive data is stored in a database, that data must be protected from unauthorized access. Oracle Retail provides the following recommendations protecting data:

- Access to the stored procedures used in the data purge scripts should be restricted.
- Authentication to the database should be done with a different user ID than authentication to the applications.

RIB does not populate the database with any pre-defined users. An administrative user is created during installation.

A

Credential Store Framework

A credential store is used for the secure storage of credentials. The credential store framework (CSF) API is used to access and perform operations on the credential store.

CSF provides the following capabilities:

- Enables the secure management of credentials.
- Provides an API for the storage, retrieval, and maintenance of credentials.
- Supports file-based, such as Oracle wallet, and LDAP-based credential management.

Oracle Retail RIB CSF Implementation

Oracle Retail Integration Bus protects authentication passwords using CSF and Oracle wallet. The credentials are stored and retrieved from the Oracle wallet store using APIs. Credential Store Manager is the utility which provides methods that can store and retrieve credentials from a wallet based file. Internally, this utility uses CSF API to manage the credentials.

B

Keytool Utility

The keytool utility is included with the JRE. It is used to create new keys, import digital certificates, export existing keys, and interact with the key management system.

Creating a Self-Signed Certificate

To create a self-signed certificate, use the following command. It creates a private key and a self-signed certificate that contains the corresponding public key:

```
keytool -genkey -keystore <keystore_location> -alias <your_alias> -keyalg RSA
```

Creating a Certificate Signing Request

To obtain a certificate signed by a real Certificate Authority, create a Certificate Signing Request:

1. Use the following command to generate the request:

   ```
   keytool -certreq -keystore <keystore_location> -alias <your_alias> -file <your_
   file.cer>
   ```

2. Once the Certificate Signing Request is saved in a file, send it to the Certificate Authority of your choice. To get a trial certificate, see the following Web site: https://www.thawte.com
3. When the response from the Certificate Authority is received, save the certificate in a file from which it can be imported. In order to import the certificate, the root certificate must be in your list of trusted certificate authorities, or you must accept the root certificate selected by the keytool utility.
4. To import the certificate, use the following command:

   ```
   keytool -import -keystore <your_keystore_name> -file <your_certificate_
   file.cer> -alias <your_alias> -trustcacerts
   ```

   For development or testing purposes, it should not be necessary to get a trial certificate or have your certificate signed.

Exporting and Importing Certificates

The server in an SSL conversation must have a private key and a certificate that verifies its identity.

- The private key is used by the server as a part of the key exchange algorithm.
- The certificate is sent to the client to identify the server. This information is obtained from the Key Store.

- The truststore is used by the client to verify the certificate that is sent by the server.

To populate the truststore with the public certificate of a server:

1. Export the RSA certificate (without the private key) from the server Key Store. For information on creating the certificate, see "Creating a Certificate Signing Request".

   ```
   keytool -export -keystore <your_keystore> -alias <your_alias> -file <your_
   file.cer>
   ```

2. Import the RSA certificate into the truststore.

   ```
   keytool -import -alias <your_alias> -keystore <your_truststore> -file <your_
   file.cer>
   ```

The certificate can be imported into any of the following files:

- cacerts, which is the default java truststore.
- jssecacerts, java truststore

C

Secure Web Services

Oracle Retail Integration Bus uses Web services for its integrations with Oracle Retail Management System. This appendix discusses security for the Web services.

WS-Security

The OASIS WS-Security specification is the open standard for Web services security. Its goal is to enable applications to secure SOAP message exchanges by providing encryption, integrity, and authentication support. WS-Security offers a general-purpose mechanism for associating security tokens with message content. The specification defines these approved token types:

- Username Token Profile
- X.509 Certificate Token Profile
- Security Assertion Markup Language (SAML) Token Profile

Web Service Security Implementation

Oracle Retail Management System Web services are protected using the WS-Security user authentication mechanism. Clients who want to access these Web services have to provide a valid user ID and password using a WS-Security Username Token.

Oracle Retail Management System Web Service

Oracle Retail Integration Bus can communicate with both secured and unsecured Oracle Retail Management System Web services. If the Web service is secured, the Oracle Retail Integration Bus adds the Username Token to the request.

Oracle Web Services Manager (OWSM) for Web Service Security

It is a new requirement for 16.0 that the Weblogic domain for RIB deployment must be a JRF domain. To create a JRF domain it is required to setup RCU.

RIB Web services applications can be secured with policyA or policyC. The Weblogic 1221 needs OWSM for Policy A and C to work. The owsm template choice while creating the weblogic domian, gives an option to deploy wsm-pm application to the admin server. The owsm policy manager (wsm-pm) is required for policies to work.

If http ports are disabled in the server, then wsm-pm app will not be reachable, unless wsm-pm is configured to use the SSL port. To configure the SSL port for wsm-pm, Oracle Enterprise Manager (EM) has to be deployed.

Refer Oracle Retail Integration Bus Installation Guide for more information on Weblogic domain creation and OWSM Policy Manager.

Overview about OWSM

OWSM provides a policy framework to manage and secure the web service applications. The OWSM agent, policy manager, and repository are the main components of the OWSM architecture. The Oracle WSM Policy Manager (wsm-pm) manages all Oracle WSM policies and needs to be running to use the Oracle WSM policy framework. OWSM policy manager reads/writes the policies, including pre-defined and custom policies from the OWSM repository.

Glossary

Glossary Term Element Structure is GlossEntry Followed by GlossTerm

The definition follows in a GlossDef element and a child Para or other appropriate element. The GlossDef element can contain informal examples, lists, and so on.

A LINE_OF_SYNTAX

```
Or you might have a line of code as part of your definition.
```