Oracle® Retail

Service-Oriented Architecture Enabler Tool Guide

Release 13.2.4

E29238-01

March 2012

Oracle Retail Service-Oriented Architecture Enabler Tool Guide, Release 13.2.4

Primary Author: Susan McKibbon

Contributing Author: David Burch

Contents

Send Us Your Comments

Oracle Retail Service-Oriented Architecture Enabler Tool Guide, Release 13.2.4

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 new Applications Release Online Documentation CD available on My Oracle Support and www.oracle.com. 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 www.oracle.com.

Preface

The *Oracle Retail Service-Oriented Architecture Enabler (RSE) Tool Guide* provides information about the tool as well as installation instructions.

Audience

The *Oracle Retail Service-Oriented Architecture Enabler (RSE) Tool Guide* is written for the following audience:

- Database administrators (DBA)
- System analysts and designers
- Integrators and implementation staff

Related Documents

For more information, see the following documents in the Oracle Retail Integration Bus 13.2.4 documentation set:

- *Oracle Retail Integration Bus Implementation Guide*
- *Oracle Retail Integration Bus Installation Guide*
- *Oracle Retail Integration Bus Operations Guide*
- *Oracle Retail Integration Bus Release Notes*
- *Oracle Retail Integration Bus Hospital Administration Guide*
- *Oracle Retail Functional Artifacts Guide*
- *Oracle Retail Functional Artifact Generator 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, 13.1) or a later patch release (for example, 13.1.2). If you are installing the base release, additional patch, and bundled hot fix releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch and bundled hot fix releases can contain critical information related to the base release, as well as information about code changes since the base release.

Oracle Retail Documentation on the Oracle Technology Network

Documentation is packaged with each Oracle Retail product release. Oracle Retail product documentation is also 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. These documents are packaged with released code, or you can obtain them through My Oracle Support.)

Documentation should be available on this Web site within a month after a product release.

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 Introduction

The purpose of the Retail Service-Oriented Architecture Enabler (RSE) tool is to provide a standard, consistent way to develop Web services for PL/SQL and Java EE applications. Because it allows them to expose their business logic, the focus of development can be on the business logic code, not on the Web service infrastructure.

The RSE tool creates Web service provider end-points, consumer clients for Web service providers, and templates for interfacing with PL/SQL APIs and Java EE APIs.

The tool also produces design time and run time artifacts. It works in conjunction with another RTG tool, the Retail Functional Artifact Generator.

Note: For more information on the tool, see the *Oracle Retail Functional Artifact Generator Guide*.

Major Features of the RSE Tool

The following is a list of the essential features of the RSE tool:

- The RSE tool is standards based.
  - All services are generated in a consistent and standard manner.
  - All services are SOAP/HTTP based Web services.
  - All services comply to the JAX-WS specification.
  - All services are WS-Addressing enabled.
  - WS-Security can be plugged into these Web services without any code change.
  - All Web services are Document Literal Wrapped.
  - Generated services are capable of using SOAP headers.
- The RSE tool generates technology-specific API templates for PL/SQL APIs and Java EE.
  - It supports PL/SQL as a Web service provider.
  - PL/SQL code can directly call any third party SOAP/HTTP based Web services.
  - It supports java code as a Web service provider.
  - It supports java code as a Web service consumer.

- Generation by the RSE tool is controlled by a single Service Definition Library XML file.
  - By creating Web services from the high level abstraction in the Service Definition Library, top down Web services development is supported.
  - All service operation inputs and outputs are validated against the XML schema.
  - There is a single source truth for all service and domain object documentation.
  - The same documentation is propagated to static WSDL, Java/PLSQL API code, UDDI published content, and live WSDL.
  - The Service Definition Library XML file is a service-oriented architecture governance asset.
- The generated services deploy in any Java EE 5 compliant application server, with certification on Oracle WebLogic Server. (Services are deployable to a clustered Java EE application server.)
- The generated services are callable as SOAP based Web services over SOAP/HTTP, local EKJB calls, remote EJB calls, or POJO services.
- All services support Web service versioning strategy.
- All generated Web services are forward and backward compatible.
- For every Web service, a static WSDL is generated. (The generated static WSDL pulls in all of the Business Object (BO) and Web service level documentation.)
- All deployed services can be published to any standard UDDI registry.
  - UDDI publishing has been tested with both WebLogicServer and Oracle Service Repository (OSR).
  - Every generated <*appname*>-service.ear contains an Infrastructure Management Service that can "talk to" the UDDI registry and publish all the services available within the.ear to the registry.
- Services can take advantage of Oracle Database Real Application Cluster (RAC).
- The RSE tool has built-in functionality.
  - Every service generated has a ping operation to test for network connectivity.
  - A Service Operation Context is passed to both Java EE and PL/SQL service provider API implementation code.
  - The Web service consumer generated has client side asynchronous service invocation capability.
  - User-defined WebService Faults are automatically generated and handled by the infrastructure at runtime. The definitions are made in the Service Definition Library XML file.
- All Web service operations are transactional. A SOAP Fault response automatically rolls back the service operations transaction. A success response automatically commits the service operations transaction.
- Web service consumers do not participate in the Web service provider side transaction. There is no transaction context propagation from client to server.

Concepts

Service-oriented architecture (SOA) is a strategy for constructing business-focused, software systems from loosely coupled, interoperable building blocks (called Services) that can be combined and reused quickly, within and between enterprises, to meet business needs (as described in Oracle Fusion Reference Architecture, SOA Foundation Release 1.0).

Service Infrastructure products focus on enabling SOA projects, rather than developing new business function, or providing for other business driven needs. The goal of Service Infrastructure is to enable the delivery teams to deliver SOA projects faster, and to make the overall SOA undertaking much more manageable.

The Retail Service-Oriented Architecture Enabler Tool (RSE) is designed and developed to support the creation of Web Services by allowing a high level abstraction, higher than the WSDL, and tailored to the business analyst/functional analyst. The Business Analyst can easily understand, define, and design without knowing the intricacies of WSDLs and the technical details of the implementation. This approach also is called top-down Web services development.

What is a Service?

A Service can be described as a way of packaging reusable software building blocks to provide functionality to users and to other services. A service is an independent, self-sufficient, functional unit of work that is discoverable, manageable, and measurable, has the ability to be versioned, and offers functionality that is required by a set of users or consumers.

A logical definition of a Service has three components:

- Contract: A description of what the service provides (and its constraints).
- Interface: The means by which the service is invoked.
- Implementation: The deployed code and configuration of infrastructure.

Oracle Fusion Reference Architecture (OFRA)

It is important to understand the position and role of the RSE tool within the broader context of service-oriented architecture and development. It is beyond the scope of this document to cover the range of SOA approaches and methodologies, but it is necessary to cover some aspects to place the tool in the appropriate context.

Oracle has developed and published the Oracle Fusion Reference Architecture (OFRA) for building and integrating enterprise-class solutions, part of the IT Strategies from Oracle collection.

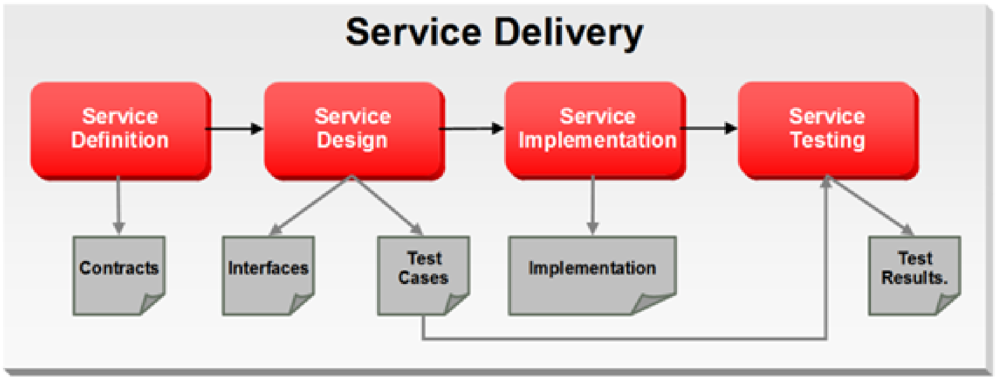
The Oracle Fusion Architecture Framework is a collection of assets designed to provide guidance on building solutions for the Oracle Fusion solution environment, which includes the Oracle Fusion Reference Architecture (OFRA). The following diagrams and definitions are from OFRA documentation.

Note: See Oracle Practitioner Guide Software Engineering in an SOA Environment Release 1.0 E14486-01.

The service analysis phase of the Oracle Service Engineering Framework consists of three main sets of engineering practices: SOA Requirements Management, Service Identification & Discovery, and Service Release Planning.

As with traditional software engineering, service engineering also begins with requirements and analysis, as illustrated below:

After Service Analysis, the next phase is Service Delivery, which includes the core delivery engineering activities. In this phase, a service candidate is molded into one or more services. Service candidates entering this phase have been justified for realization and scheduled for release.



Service Delivery begins with Service Definition, which primarily determines service boundaries as well as the construction of the service contract.

Service Design then acts upon the Service contracts to develop the Services' interfaces. The process of defining a Service interface is much more involved than simply coming up with the input and output for the Service. Service design analyzes the contract from the consumer's perspective, and is influenced by factors such as scope (enterprise, LOB, application, and so on), message exchange patterns (MEPs) as well as non-functional requirements such as expected volume, and response time requirements (specified in the contract).

Service Implementation ensures that all aspects of the Service contracts are implemented and upheld through the delivery of business logic as well as the deployment to Service Infrastructure. The implementation must faithfully realize the Service Contract and interface which are defined through Service definition and design.

Note: See: Oracle Fusion Reference Architecture, Overview. Release 1.0 E14482-01

Where Does RSE Fit?

The Retail Service-Oriented Architecture Enabler (RSE) is a Service Infrastructure tool developed by Oracle Retail to enable the adoption of service-oriented architecture (SOA) and avoid some of the typical pitfalls of many SOA projects. It addresses many common issues, such as versioning, contract design, security, consistency, reuse, documentation, governance, compliance, and customization. It does this by enforcing SOA Best Practices and patterns that are proven and time tested by various SOA pioneers.

The tool provides the capability for business analysts and developers to define the correct service contract. It provides ease-of-use and a level of abstraction such that the domain experts or subject matter experts are not required to understand code to design services. The SOA developers can be completely focused on implementing the business logic code behind the service and do not have to worry about SOA infrastructure issues such as versioning and customization.

The Retail Service-Oriented Architecture Enabler Tool fits within the Service Delivery phases. The appropriate use of the tool is after the service analysis phases and the development team is ready for service definition and design. The RSE tool outputs can then be used in the Service Implementation.

RSE is designed to support this type of approach, which also is called top-down Web services development.

Technical Specifications

The Oracle Retail SOA Enabler tool has dependencies on Oracle Retail application installations. This section covers these requirements.

Supported Operating Systems

Supported On	Version Supported
Oracle WebLogic Server OS	OS certified with OracleWebLogic Server 11 g Release 3 (10.3.3). Options are AIX 6.1 and Oracle Linux 5 update 5.
Oracle WebLogic Server	Oracle WebLogic Server 11g Release 3 (10.3.3).

2

Installation and Basic Setup

This chapter explains how to deploy the Retail Service-Oriented Architecture Enabler tool to an Oracle WebLogic application server as a Web application.

Installation as a Web Application in Oracle WebLogic

The steps below describe how to deploy the Retail Service-Oriented Architecture Enabler tool to an Oracle WebLogic Application Server as a Web application.

Note: See "Technical Specifications" in Chapter 1.

Prerequisites

The following are prerequisites for installation.

- The retail-soa-enabler-gui.war file is located within the directory structure of the RetailSOAEnabler13.2.4ForAll13.2.4Apps_eng_ga.tar. Locate and extract the contents to a location that is accessible by the browser for deployment.
- The installation and base configuration of the Oracle WebLogic Server is beyond the scope of this document. Work with the Application Server Administration team to determine the physical and logical placement of the retail-soa-enabler-gui component within the WebLogic Server deployment.

Note: See the *Oracle WebLogic Server 11g Release 3 (10.3.3) Installation Guide.*

Deploy the Retail Service-Oriented Architecture Enabler

Using the WebLogic Server Administration Console, complete the following steps:

Note: For instructions with illustrations (screen captures), see "Appendix: Installer Screens."

1. Navigate to the Deployments page.
2. If necessary, click **Lock and Edit** on the left navigation bar to enable the Install button.
3. Click **Install**.

   Note: If the application has already been installed, see "Redeploy the Application".

   The **Locate deployment to install and prepare for deployment** page is displayed. Follow the instructions to locate the retail-soa-enabler-gui.war file.
4. Select **Upload Files**.
5. On the **Upload a Deployment to the admin server** page, use the Browse button to locate the retail-soa-enabler-gui.war file in the Deployment Archive.
6. Select the retail-soa-enabler-gui.war.
7. Click **Next** and move to **Choose targeting style**.
8. Select **Install this deployment as an application**.
9. Click **Next** and move to Optional Settings.
10. Click **Next** and move to **Review your choices and click Finish**.
11. Select **No, I will review the configuration later**.
12. Click **Finish** to deploy the application.

Verify the Retail Service-Oriented Architecture Enabler

1. Navigate to the Deployments page.
2. Locate retail-soa-enabler-gui on the Summary of Deployments page.
3. Click the name, **retail-soa-enabler-gui**, to move to the **Settings for the retail-soa-enabler-gui**.
4. Select the **Testing** tab.
5. Click the **index.jsp URL** in the Test Point.
6. The URL should open to the Retail Service-Oriented Architecture Enabler Home page.
7. The installation is complete. See Chapter 4, "User Interface Usage."

Redeploy the Application

If the retail-soa-enabler-gui application has already been deployed, follow these steps:

1. If the retail-soa-enabler-gui application is running, select **Stop** and **When Work Completes** or **Force Stop Now**, depending on the environment. The recommended option always is **When Work Completes**.
2. Select **Delete**.
3. The retail-soa-enabler-gui should now not show on the Summary of Deployment page.
4. Return to the appropriate step in "Deploy the Retail Service-Oriented Architecture Enabler."

3 Tool Inputs and Outputs

This chapter describes the tool inputs and tool outputs associated with RSE.

Tool Inputs

Tool inputs include the following:

- ServiceProviderDefLibrary.xml
- XSDs and retail-public-payload-java-beans.jar
- PL/SQL Oracle Objects
- WSDL

ServiceProviderDefLibrary.xml

This is the definition file for Provider services for both PL/SQL and Java EE services, and conforms to the ServiceProviderDefLibrary.xsd schema. This definition file contains a high level definition of a set of services which use Retail Business Objects (BOs) as inputs and outputs.

XSDs and retail-public-payload-java-beans.jar

- The RSE tool references JAXB created java beans based on the BO source schema XSDs. These beans are contained in the retail-public-payload-java-beans.jar.
- The RSE tool will use Oracle Retail BOs from retail-public-payload-java-beans.jar and custom BOs from custom-retail-public-payload-java-beans.jar.
- The jar file is located in the WebLogic deployment directory where the RSE tool is deployed.
- The jar file is created using the Retail Artifact Generator from the source BO XSDs.
- The jar file also contains the source XSDs themselves, which will be used by the deployed service to validate all requests and responses against.

PL/SQL Oracle Objects

These are artifacts that are created from the XSDs using the Retail Artifact Generator. The Objects have to installed into the database and accessible to the target Web service APIs generated by RSE.

WSDL

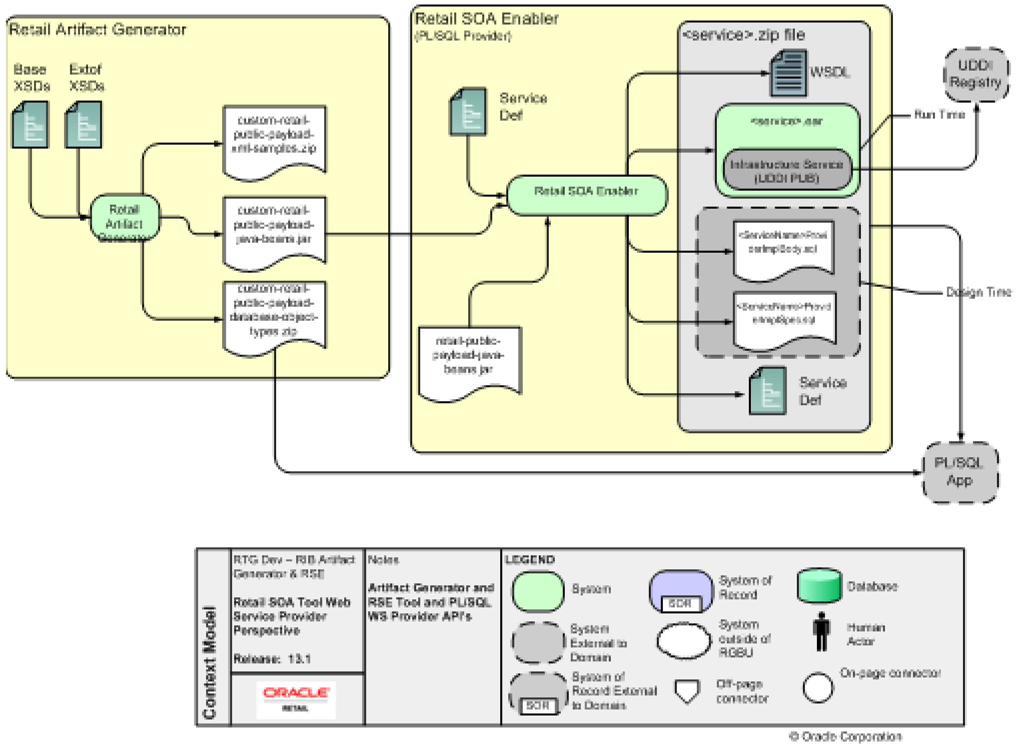
For the Web service consumers, the input is the WSDL of the Web service provider that the service will be consuming.

Tool Outputs

Tool outputs include the following:

- PL/SQL Provider Web Service
- PL/SQL Consumer Web Service
- Java EE Provider Web Service
- Java EE Consumer Web Service

PL/SQL Provider Web Service


Retail Artifact Generator
Base XSDs
Extof XSDs
Retail Artifact Generator
custom-retail-public-payload-xml-samples.zip
custom-retail-public-payload-java-beans.jar
custom-retail-public-payload-database-object-types.zip
Retail SOA Enabler
(PL/SQL Provider)
Service Def
Retail SOA Enabler
retail-public-payload-java-beans.jar
<service>.zip file
WSDL
<service>.ear
Infrastructure Service (UDDI PUB)
<ServiceName>ProviderImplBody.sql
<ServiceName>ProviderImplSpec.sql
Service Def
UDDI Registry
Run Time
Design Time
PL/SQL App
Context Model
RTG Dev – RIB Artifact Generator & RSE
Retail SOA Tool Web Service Provider Perspective
Release: 13.1
ORACLE RETAIL
Notes
Artifact Generator and RSE Tool and PL/SQL WS Provider APIs
LEGEND
System
System of Record
SOR
Database
System External to Domain
System outside of RGBU
Human Actor
System of Record External to Domain
Off-page connector
On-page connector
© Oracle Corporation

PL/SQL Applications (such as the Oracle Retail Merchandising System) use Oracle Objects, which are similar to the Oracle Retail RIB style APIs. The tool generates a Web service provider layer between the external clients and the PL/SQL APIs to provide the Web service functionality.

The RSE PL/SQL provider output is a zip file. The filename convention is <app>_PLSQLServiceProvider.zip. For example, rms_PLSQLServiceProvider.zip. The zip file contains the following:

- <*ServiceName*>ProviderImplSpec.sql

  This is the specification for the <*ServiceName*>. It creates the package for the <*ServiceName*> in the <*app*> database. It describes all the operations and their IN and OUT parameters for the service.

- <*ServiceName*>ProviderImplBody.sql

  This is the package body for the <*ServiceName*>. This is where the application teams have to write the business logic.

- <*app*>-service.ear

  The.ear file has to be deployed on an Oracle WebLogic. The steps for deployment are in the RSE PL/SQL WS Installation Guide.

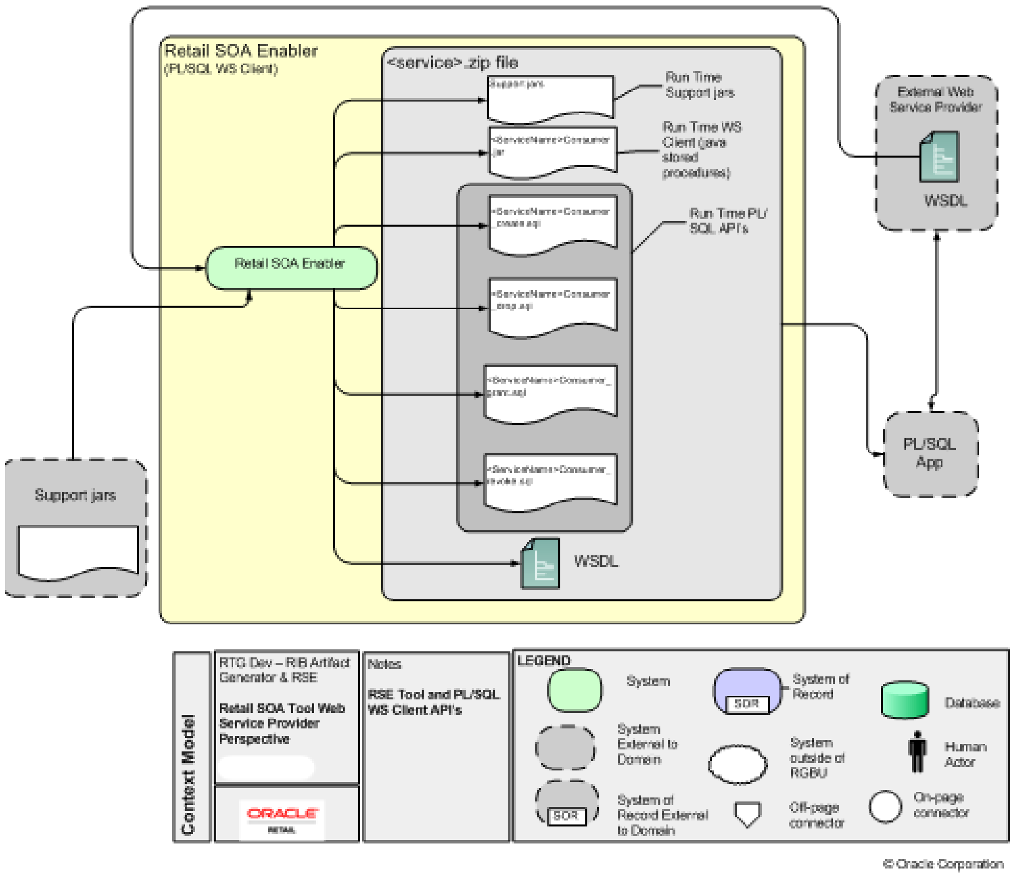
- ServiceProviderDefLibrary.xml

  This is a copy of the ServiceProviderDefLibrary.xml file that was used to create the output.

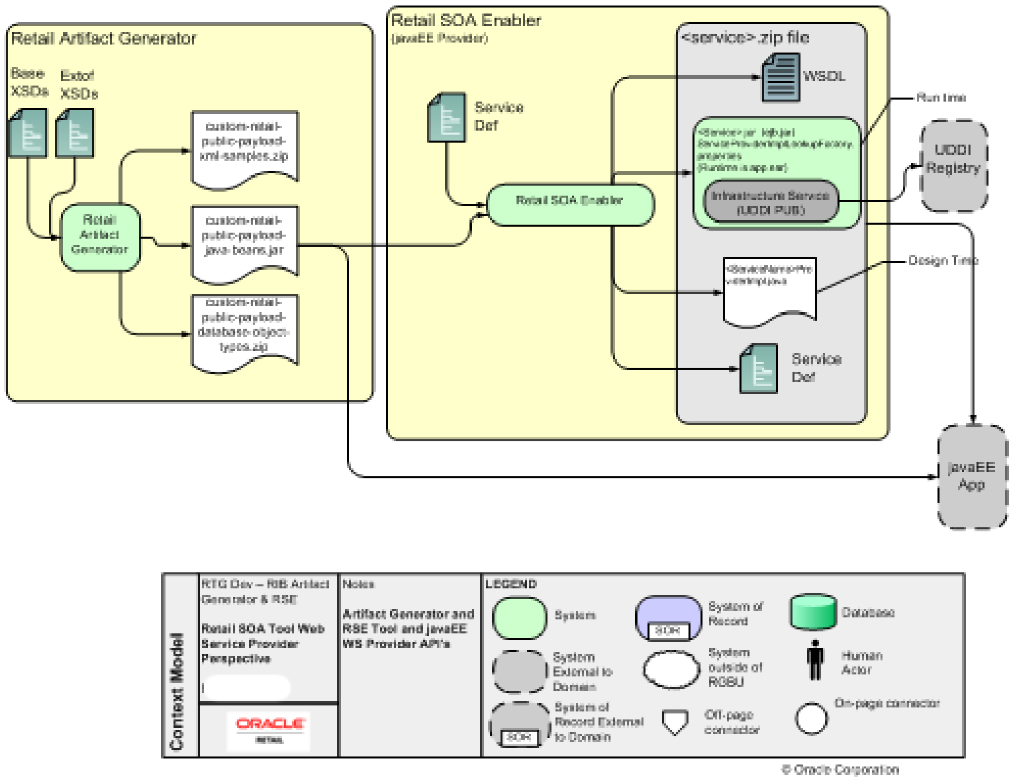
- <*ServiceName*>Service.wsdl

  This is a WSDL file describing the generated Web service. This WSDL file will be fully documented, pulling in documentation elements from both the service def file as well as the BO XSD files. This is a single file with all types inlined. It can be used as input to create a consumer for the generated provider.

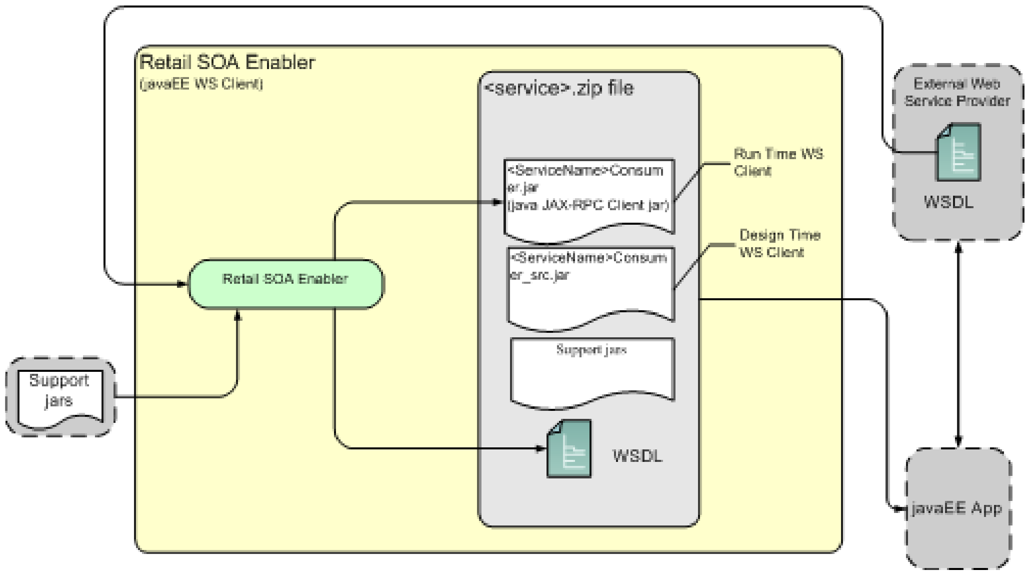
PL/SQL Consumer Web Service

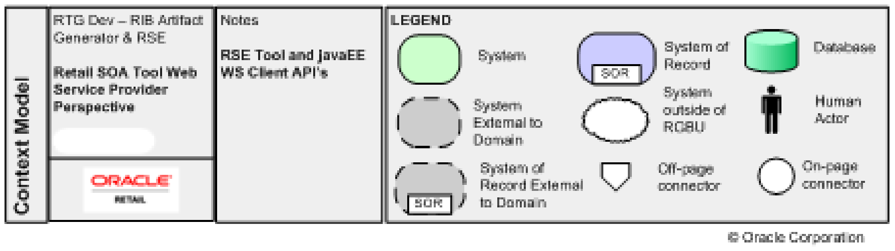

Retail SOA Enabler
(PL/SQL WS Client)
<service>.zip file
Support jars
Run Time Support jars
<ServiceName>Consumer.jar
Run Time WS Client (java stored procedures)
<ServiceName>Consumer_create.sql
Run Time PL/SQL API's
<ServiceName>Consumer_drop.sql
<ServiceName>Consumer_grant.sql
<ServiceName>Consumer_invoke.sql
WSDL
Retail SOA Enabler
Support jars
External Web Service Provider
WSDL
PL/SQL App
Context Model
RTG Dev – RIB Artifact Generator & RSE
Retail SOA Tool Web Service Provider Perspective
Notes
RSE Tool and PL/SQL WS Client API's
LEGEND
System
System of Record
SOR
Database
System External to Domain
System outside of RGBU
Human Actor
System of Record External to Domain
Off-page connector
On-page connector
© Oracle Corporation

Java EE Provider Web Service


Retail Artifact Generator
Base XSDs
Extof XSDs
Retail Artifact Generator
custom-retail-public-payload-xml-samples.zip
custom-retail-public-payload-java-beans.jar
custom-retail-public-payload-database-object-types.zip
Retail SOA Enabler
(javaEE Provider)
Service Def
Retail SOA Enabler
<service>.zip file
WSDL
<Service>.jar, lejb.jar
ServiceProviderImplLookupFactory.properties
(Runtime is app.ear)
Infrastructure Service (UDDI PUB)
<ServiceName>ProviderImpl.java
Service Def
Run time
Design Time
UDDI Registry
javaEE App
Context Model
RTG Dev – RIB Artifact Generator & RSE
Retail SOA Tool Web Service Provider Perspective
ORACLE RETAIL
Notes
Artifact Generator and RSE Tool and javaEE WS Provider API's
LEGEND
System
System of Record
SOR
Database
System External to Domain
System outside of RGBU
Human Actor
System of Record External to Domain
Off-page connector
On-page connector
© Oracle Corporation

Java EE Consumer Web Service




Context Model
RTG Dev – RIB Artifact Generator & RSE
Retail SOA Tool Web Service Provider Perspective
ORACLE RETAIL
Notes
RSE Tool and javaEE WS Client API's
LEGEND
System
System of Record
SOR
Database
System External to Domain
System outside of RGBU
Human Actor
System of Record External to Domain
SOR
Off-page connector
On-page connector
© Oracle Corporation

4 User Interface Usage

The Retail Service-Oriented Architecture Enabler (RSE) tool produces design time and run time artifacts, and it works in conjunction with another tool, the Retail Functional Artifact Generator.

Note: See the *Retail Functional Artifact Generator Guide*.

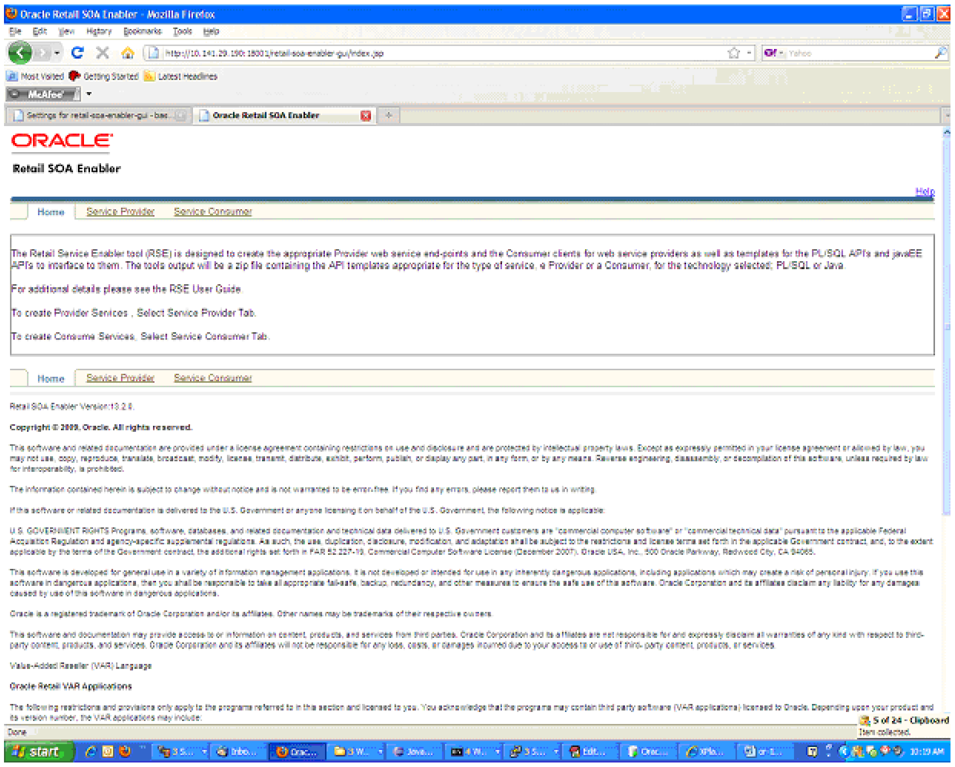
The graphical user interface (GUI) for RSE is hosted on an Oracle WebLogic server as a Web application. Once installed and configured, the GUI is accessed through a URL (http://host:port/contextroot). For example, http://linux1:7001/retail-soa-enabler-gui.

The RSE user interface has three tabs, or sections:

- Home
- Server Provider
- Service Consumer

  The user interface is designed to be easy to use. Online help is available, including examples for each function.

The following is the Home Page.


Oracle Retail SOA Enabler - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://10.141.29.196:18001/retail-soa-enabler-gui/index.jsp
Most Visited Getting Started Latest Headlines
McAfee
Settings for retail-soa-enabler-gui - bas...
Oracle Retail SOA Enabler
ORACLE
Retail SOA Enabler
Help
Home
Service Provider
Service Consumer

The Retail Service Enabler tool (RSE) is designed to create the appropriate Provider web service end-points and the Consumer clients for web service providers as well as templates for the PL/SQL API's and javaEE API's to interface to them. The tools output will be a zip file containing the API templates appropriate for the type of service, a Provider or a Consumer, for the technology selected; PL/SQL or Java.

For additional details please see the RSE User Guide.

To create Provider Services , Select Service Provider Tab.

To create Consume Services, Select Service Consumer Tab.

Retail SOA Enabler Version:13.2.0.

Copyright © 2009, Oracle. All rights reserved.

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

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

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

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

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third- party content, products, or services.

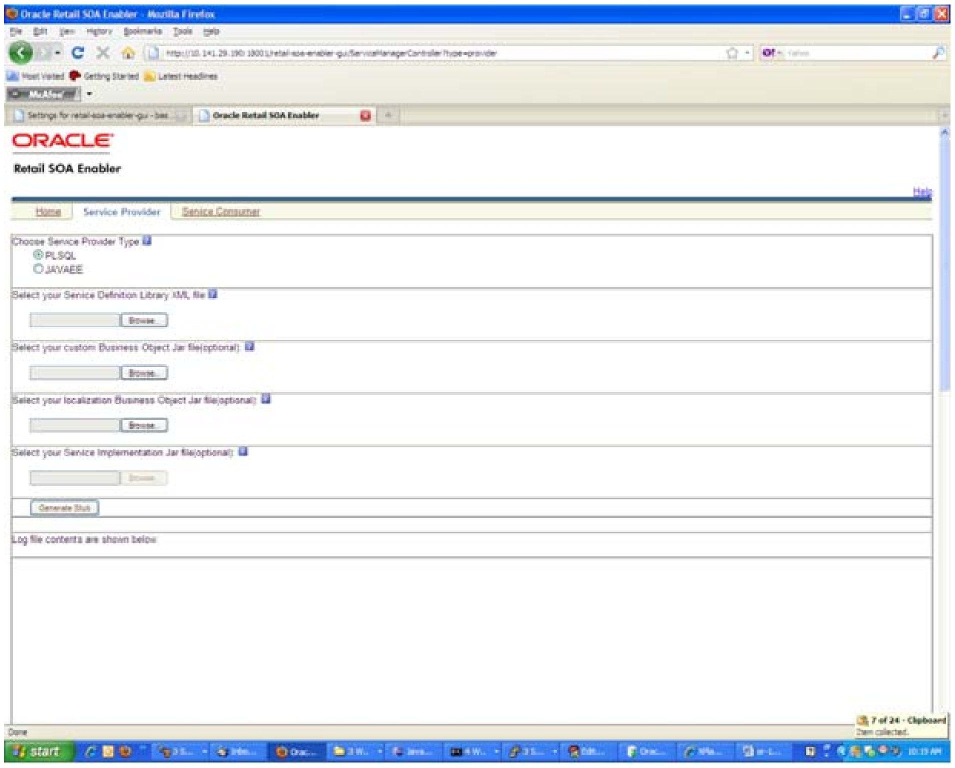
Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

Service Provider

The service provider screen gives the option of selecting the Provider type (a Java EE or a PL/SQL service provider).


Oracle Retail SOA Enabler - Mozilla Firefox
File Edit View History Bookmarks Tools Help
ORACLE
Retail SOA Enabler
Help
Home
Service Provider
Service Consumer
Browse...
Generate Stub

Choose Service Provider Type
[✓ PLSQL] JAVAEE

Select your Service Definition Library XML file

Select your custom Business Object Jar file(optional):

Select your localization Business Object Jar file(optional):

Select your Service Implementation Jar file(optional):

Log file contents are shown below

A PL/SQL service provider can be used by PL/SQL applications such as RMS to expose PL/SQL packages as Web services. The Java EE service provider option allows Java EE applications to create Web services using Oracle Retail payload classes as input and outputs.

The generated Web services do not have any business logic in them. They provide only the framework for the development of Web services.

The inputs for creating Java EE or PL/SQL Web services are as follows.

- Service Definition Library XML file
- Custom Business Objects jar file
- Service Implementation jar file

Service Definition Library XML File

The mandatory input for creating a Java EE or a PL/SQL service provider is a Service Definition Library XML file. This file should contain all the details about the Web services that need to be created.

Note: See Chapter 5," Service Definition Library XML File."

Custom Business Objects Jar File

While creating Web services, users may want to use their own payloads extend the existing payloads. These payloads are known as custom payloads and can be provided to the tool as an input for creating Web services. The service provider screen has a field for custom Business Objects jar file. It allows the user to upload a jar file which contains the custom payloads. This jar file is optional; if this is not provided the base payloads are used to create the Web services.

Note: See the *Oracle Retail Functional Artifact Generator Guide* for how to create a custom Business Objects jar file.

Localization Busines Object Jar File

While creating Web services, users may want to use localized version of payloads. These payloads are known as localized payloads and can be provided to the tool as an input for creating Web services. The service provider screen has a field for localization Business Object Jar file. It allows the user to upload a jar file which contains the localized payloads. This jar file is optional; if this is not provided, the base payloads are used to create the Web services.

Note: See the Oracle Retail Functional Artifacts Generator Guide for how to create a localization Business Objects jar file.

Service Implementation Jar File

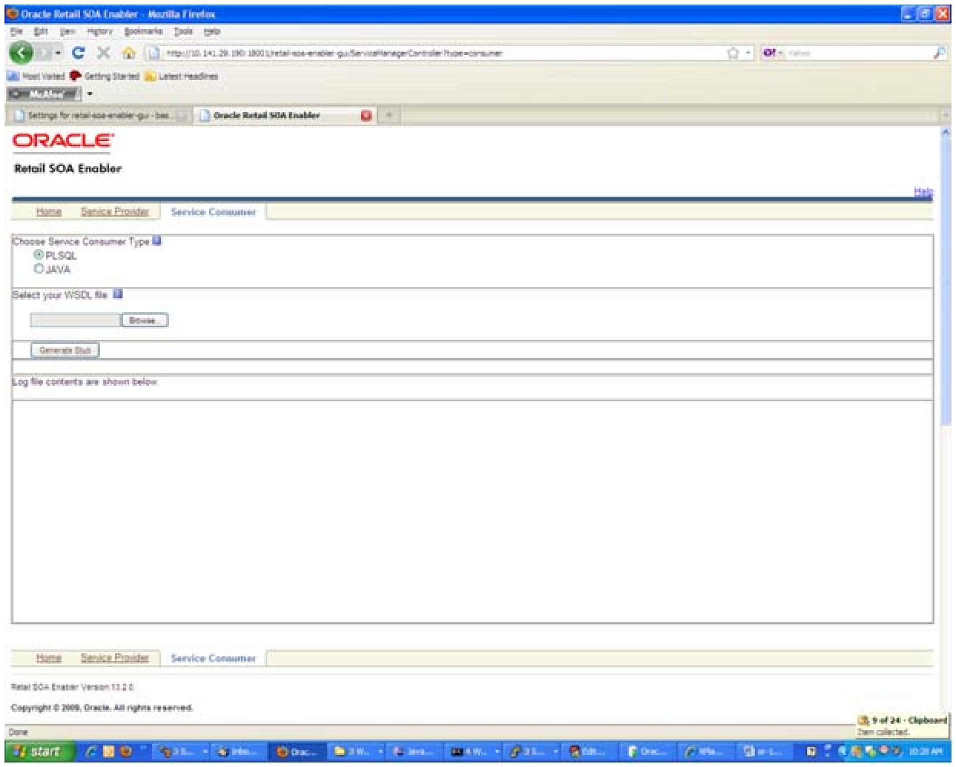
This jar file is used only while creating Java EE Web services. While creating Java EE Web services the tool generates empty implementation for the services. Users will have to create their own implementation classes for the Web services and use those classes in the generation of the .ear file in the zip file.

After entering the file names in all the text boxes, click **Generate Stub**. This generates a .zip file with an .ear file, which is deployed to a WebLogic server.

Note: See Chapter 7, "Creating the Java EE Implementation Jar."

Service Consumer

The Service Consumer tab allows for the creation of a Java EE or PL/SWL service consumer. After an input WSDL file is selected, the tool runs. When the tool is finished, the consumer distribution zip file can be downloaded to a specific location.


Oracle Retail SOA Enabler - Mozilla Firefox
ORACLE
Retail SOA Enabler
Help
Home
Service Provider
Service Consumer
Choose Service Consumer Type
PLSQL
JAVA
Select your WSDL file
Browse...
Generate Stub
Log file contents are shown below
Retail SOA Enabler Version 13.2.0
Copyright © 2009, Oracle. All rights reserved.

Help

Click the Help link on the right upper corner of the Home page for a brief description of the Service Provider and Service Consumer functionality.

5 Service Definition Library XML File

The Service Definition Library SML file (ServiceDef) is the mandatory input for creating a Java EE or a PL/SQL service provider. This file should contain all the details about the Web services that need to be created.

This chapter provides a detailed description of each section of the schema as well as instructions for managing the Service Definition Library XML file.

Schema Definition

This section discusses the elements of the schema, beginning with the root element and including child elements.

serviceProviderDefLibrary

This is the root element of the schema. The following is an example of the serviceProviderDefLibrary element:

```
<xs:element name="serviceProviderDefLibrary">
       <xs:complexType>
              <xs:sequence>
                     <xs:element ref="service" maxOccurs="unbounded" />
              </xs:sequence>
              <xs:attribute name="appName" type="xs:string" use="required"/>
    <xs:attribute name="version" type="xs:string" use="optional" default="v1"/>
    <xs:attribute name="serviceNamespacePattern" type="xs:string" use="optional"
default="http://www.oracle.com/retail/APPNAME/integration/services/SERVICENAMEServ
ice/VERSION"/>
       </xs:complexType>
</xs:element>
```

Attributes

The serviceProviderDefLibrary has the following attributes:

- **appName**

  This is the name of the application for which the. ear file is being built. When the .ear file is generated, the name of the .ear file starts with the application name. The format of the generated .ear file is <*appName*>-service.ear. For example, if the *appName* is **rms**, the .ear file name is rms-service.ear.

- **serviceNamespacePatter**

  This attribute specifies the pattern for the namespaces that are generated for the Web services. The default value for this attribute is http://www.oracle.com/retail/APPNAME/integration/services/SERVICENAMEService/VERSION.

- **Version**

  This is the version of the service definition.

Elements

The serviceProviderDefLibrary contains the following elements:

service

Each service element in serviceProviderDefLibrary represents one Web service. The service provider definition should have at least one service defined in it.

The following is an example of the service element:

```
<xs:element name="service">
       <xs:complexType>
              <xs:sequence>
                    <xs:element ref="documentation" minOccurs="0" />
                    <xs:element ref="operation" maxOccurs="unbounded" />
              </xs:sequence>
              <xs:attribute name="name" type="xs:string" use="required"/>
              <xs:attribute name="serviceNamespace" type="xs:string"
use="optional"/>
              <xs:attribute name="serviceVersion" type="xs:string" use="optional"
default="v1"/>
              <xs:attribute name="custom" type="xs:boolean" use="optional"
default="false"/>
      </xs:complexType>
</xs:element>
```

The service element has the following attributes:

- name

  This is the name of the Web service to be created.

- serviceNamespace

  This is the namespace in which the Web service will be created.

- serviceVersion

  This is the version of the Web service. The default value is v1.

- custom

  This attribute specifies whether the service is a custom service. A custom service uses custom payload as input or output for any of its methods.

The service element contains the following elements:

- Documentation

  This field describes the purpose of the service.

- Operation

  The operation represents the method in the generated Web service. Each service should contain at least one operation.

  The following is an example of the operation element:

  ```
  <xs:element name="operation">
         <xs:complexType>
                <xs:sequence>
                       <xs:element ref="documentation" minOccurs="0" />
                       <xs:element ref="input" />
                       <xs:element ref="output" minOccurs="0" />
                       <xs:element ref="fault" minOccurs="0"
                             maxOccurs="unbounded" />
               </xs:sequence>
               <xs:attribute name="name" type="xs:string" use="required"/>
               <xs:attribute name="suffix" default="inputType">
              <xs:simpleType>
                    <xs:restriction base="xs:string">
                          <xs:enumeration value="inputType" />
                          <xs:enumeration value="outputType" />
                          <xs:enumeration value="NONE" />
              </xs:restriction>
       </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="custom" type="xs:boolean" use="optional" default="false"/>
  </xs:complexType>
  </xs:element>
  ```

The operation element has the following attributes:

- name

  This is the name of the operation.

- suffix

  This is the string to be added to the end of the operation name. One of the following values are supported for this attribute:

  – inputType

    If the suffix value is inputType, the input type name of the operation is added to the generated method name. For example, if the operation name is **create** and input type for that operation name is SupplierDesc, the generated operation name will be createSupplierDesc.

  – outputType

    If the suffix value is outputType, the output type name of the operation is added to the generated method name. For example, if the operation name is **create** and output type for that operation name is SupplierRef, the generated operation name will be createSupplierRef.

   - NONE

     If the suffix value is NONE, a suffix is not added to the operation name.

     Note: If no value is provided for the suffix attribute, inputType is used as the default value.

- custom

  If the operation is custom, this attribute should be set to true. The operation is considered custom if it uses a custom payload for input or output.

The operation element contains the following child elements.

- Documentation
- Input
- Output
- Fault

  Fault contains the following elements:

  - Documentation

    The description of the fault.

  - Faulttype

    The name of the fault.

Managing the Service Definition Library XML File

The Service Definition Library XML file is the single source of truth for the RSE tool. This section discusses the creation and management of the file.

Creating the File

The Service Definition Library XML example in "Appendix: Sample ServiceProviderDefLibrary.xml" can be used as the initial template. Use the instructions in the Service Definition Library XML File section to construct the ServiceDef according to the goals of the Service requirements.

As discussed in the Concepts section, the creation of this file is the result of the analysis phase and part of the Service Design phase. The template provides the placeholders for the standard Service components: Service name, operation name, and the contracts for each of the operations, as well as the standard faults.

The ServiceDef should be created and managed (or governed) as a service-oriented architecture asset in a source code control system. It is as important as the Service Contracts (XSDS) and implementation source code.

Changing the Version of the File

To change the version of the service definition libraryfile, a **version** attribute must be added to the root element, serviceProviderDefLibrary.

For example:

```
<serviceProviderDefLibrary appName="rms"
xmlns=http://www.oracle.com/retail/integration/services/serviceProviderDefLibrary/
v1
version="v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...

</serviceProviderDefLibrary>
```

Changing the appName Attribute in the File

To change the application name in the services, edit the *appName* attribute in the root element, serviceProviderDefLibrary.

For example:

```
<serviceProviderDefLibrary appName="editThisAppName"
xmlns=http://www.oracle.com/retail/integration/services/serviceProviderDefLibrary/
v1 version="v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...

</serviceProviderDefLibrary>
```

Renaming a Service or Operation Name in the File

To rename a service, edit the name attribute in the service element.

For example:

```
<serviceProviderDefLibrary appName="rms"
xmlns=http://www.oracle.com/retail/integration/services/serviceProviderDefLibrary/
v1
version="v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<service name="EditThisName">
...
</serviceProviderDefLibrary>
```

To rename an operation in the service, edit the name attribute of the other operation element.

Adding a New Service or New Operation to the File

To add a new service to library, add a new service element with its child elements.

For example:

```
<serviceProviderDefLibrary appName="rmscostchange"
xmlns=http://www.oracle.com/retail/integration/services/serviceProviderDefLibrary/
v1
version="v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
      <service name="ExistingService">
              <operation name="existingOperation">
                     <documentation></documentation>
                     <input type="XXX">
                             <documentation></documentation>
                     </input>
                     <output type="YYY">
                            <documentation></documentation>
                     </output>
                     <fault faultType="IllegalArgumentWSFaultException">
                             <documentation>Throw this exception when a
"soap:Client" side message problem occurs.</documentation>
                      </fault>
                      <fault faultType="EntityAlreadyExistsWSFaultException">
                             <documentation>Throw this exception when the attempt
made to  create a object that already exists.</documentation>
                       </fault>
                       <fault faultType="IllegalStateWSFaultException">
                               <documentation>Throw this exception when an unknown
             "soap:Server" side problem ccurs.</documentation>
                       </fault>
             </operation>
</service>
<service name="AddedNewServiceName">
             <operation name="Operation">
                    <documentation></documentation>
                    <input type="XXX">
                           <documentation></documentation>
                    </input>
                    <output type="YYY">
                           <documentation></documentation>
                    </output>
                    <fault faultType="IllegalArgumentWSFaultException">
                            <documentation>Throw this exception when a
soap:Client" side message problem occurs.</documentation>
                     </fault>
                     <fault faultType="EntityAlreadyExistsWSFaultException">
                             <documentation>Throw this exception when the attempt
made to create a object that already exists.</documentation>
                     </fault>
                     <fault faultType="IllegalStateWSFaultException">
                             <documentation>Throw this exception when an unknown
                                   "soap:Server" side problem
occurs.</documentation>
                    </fault>
             </operation>
</service>


...

</serviceProviderDefLibrary>
```

To add a new operation to a service, add the operation element with its child elements.

For example:

```
<service name="service">
              <service name="ServiceName">
              <operation name="NewAddedOperation">
                     <documentation></documentation>
                     <input type="XXX">
                            <documentation></documentation>
                     </input>
                     <output type="YYY">
                            <documentation></documentation>
                     </output>
                     <fault faultType="IllegalArgumentWSFaultException">
                             <documentation>Throw this exception when a
"soap:Client" side message problem occurs.</documentation>
                     </fault>
                     <fault faultType="EntityAlreadyExistsWSFaultException">
                            <documentation>Throw this exception when the attempt
made to create a object that already exists.</documentation>
                     </fault>
                     <fault faultType="IllegalStateWSFaultException">
                             <documentation>Throw this exception when an unknown
"soap:Server" side problem ccurs.</documentation>
                     </fault>
              </operation>
              <operation name="ExistingOperation">
                    <documentation></documentation>
                    <input type="XXX">
                           <documentation></documentation>
                    </input>
                    <output type="YYY">
                           <documentation></documentation>
                    </output>
                    <fault faultType="IllegalArgumentWSFaultException">
                            <documentation>Throw this exception when a
soap:Client" side message problem occurs.</documentation>
                    </fault>
                    <fault faultType="EntityAlreadyExistsWSFaultException">
                            <documentation>Throw this exception when the attempt
made to create a object that already exists.</documentation>
                       </fault>
                       <fault faultType="IllegalStateWSFaultException">
                               <documentation>Throw this exception when an unknown
                                     "soap:Server" side problem
occurs.</documentation>
                      </fault>
              </operation>
</service>
```

Deleting a Service or Deleting Operations from the File

To delete a service from the library, remove the service element and all its child elements from the library.

To delete an operation from the service, delete the operation element and all its child elements.

Defining New Exceptions to the Operations

Users can define a new exception in the service definition library. The RSE tool creates the artifacts with this new exception.

For example:

```
<operation name="ExistingOperation">
                    <documentation></documentation>
                    <input type="XXX">
                           <documentation></documentation>
                    </input>
                    <output type="YYY">
                           <documentation></documentation>
                    </output>
                    <fault faultType="IllegalArgumentWSFaultException">
                            <documentation>Throw this exception when a
"soap:Client" side  message problem occurs.</documentation>
                    </fault>
                    <fault faultType="EntityAlreadyExistsWSFaultException">
                            <documentation>Throw this exception when the attempt
made to create a object that already exists.</documentation>
                    </fault>
                    <fault faultType="IllegalStateWSFaultException">
                            <documentation>Throw this exception when an unknown
"soap:Server" side problem  occurs.</documentation>
                    </fault>
                    <fault faultType="UserDefinedException">
                            <documentation>This is user defined exception for a
particular scenerio.</documentation>
                    </fault>
             </operation>
```

Using Different Versions of Objects as Input/Output to an Operation

The version difference between objects does not impact the RSE tool, as long as the objects adhere to standards.

6

Web Service Standards and Conventions

This chapter includes standards and conventions for Web service naming and versioning.

Web Service Naming

The following standards and conventions apply to the naming of Web Services.

The Web service name should be a business noun, concept or process.

Item	Description
Recommendation	The Web service name should be a business noun, a business concept, or a business process.
Rationale	To be in alignment with other Web service standards.
Example	Supplier Service

Avoid verbs when naming Web services.

Item	Description
Recommendation	The Web service name should be a business noun, a business concept, or a business process.
Rationale	Verbs generally are at the operation level, not at the service level.
Example	Avoid names such as CreateSupplierService.

The first 30 characters of the Web service name must be unique.

Item	Description
Recommendation	The first 30 characters of the Web service name must be unique.
Rationale	Some systems truncate names at 30 characters.
Example	N/A

The integration/services qualifier should be in the namespace.

Item	Description
Recommendation	The integration/services qualifier should be in the namespace.
Rationale	
Example	http://www.oracle.com/retail/rms/integration/services/PayTerm Service.

The Web service namespace should contain the application short name.

Item	Description
Recommendation	The Web service namespace should contain the application short name.
Rationale	Multiple applications may publish services with similar names. To categorize and identify which application is hosting what service, the service namespace should contain the application short name.
Example	http://www.oracle.com/retail/rms/integration/services/PayTerm Service.

The Web service type should be document/literal wrapped.

Item	Description
Recommendation	The Web service type should be **document/literal wrapped**.
Rationale	This is defined in the WSDL.
Example	`<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>` `<operation name="createPayTermBO">` `<ns21:PolicyReference xmlns:ns21="http://www.w3.org/ns/ws-policy" URI="#PayTermServicePortBinding_createPayTermBO_WSAT_Policy"/>` `<soap:operation soapAction=""/>` `<input>` `<soap:body use="literal"/>` `</input>`

The Web service must comply with Web Service Basic Profile 1.1.

Item	Description
Recommendation	The Web service must comply with Web Service Basic Profile 1.1.
Rationale	The specification is called the WS-I Basic Profile 1.1. It consists of a set of non-proprietary Web services specifications, clarifications, refinements, interpretations, and amplifications of those specifications which promote interoperability.
Example	N/A

The Web service operation naming pattern should be verb<TopLevelComplexType>(TopLevelComplexType variable).

Item	Description
Recommendation	The operation name pattern should be either of the following: ■ verb<TopLevelComplexType>(TopLevelComplexType variable) ■ verb<NonTopLevelComplexType>Using<TopLevelComplexType> (TopLevelComplexType variable).
Rationale	The operation name should reflect the Top Level Complex Type of the service's primary entity object to ensure the name is unambiguous.
Example	createItemListBO

Web Service Versioning

Service versioning is in the namespace, including the application and the version identifier.

The service namespace is versioned.

Item	Description
Recommendation	The WSDL for the RBS will have the namespace versioned.
Rationale	For breaking changes only, the WSDL for the RBS will have the namespace versioned. http://www.oracle.com/retail/<retail app>/integration/services/<service name>/V<incremental change number>
Example	http://www.oracle.com/retail/rms/integration/services/PayTerm Service/V2

7

Creating the Java EE Implementation Jar

Creating Web services with different implementations is a three-step process, as described below.

Note: For creating an implementation class, interface classes are required.

Step 1: Generate Web Services with Default Implementation

Generate the Web services with the default implementation as follows:

1. Provide the Service Definition Library XML file and click **Generate Stub** to create a zip file.
2. The zip file contains a jar file with the interface classes for the Web services. The name pattern of the jar file is <*appName*>-service-ejb.jar.

   For example, if the application name in ServiceDef is rms, the jar file name is rms-service-ejb.jar.

   The jar file also contains a properties file named ServiceProviderImplLookupFactory.properties. This file contains the name of the Web service interface and the class implementing the Web service.

Step 2: Implement Interfaces

Implement the interfaces and create the implementation classes. The classes can be packaged in a jar file. Upload the jar file while creating the final ear file.

Step 3: Upload the jar

When using the Service Implementation Jar File option to upload the jar containing the implementations, the default service implementation jar is not included in the .ear file. Rather, the jar file provided by the user is included. When the Web service is invoked, the service implementation provided by the user is invoked.

8 Implementation Guidelines

This chapter provides a set of implementation notes that may be helpful when implementing the Oracle Retail Service-Oriented Architecture Enabler (RSE) tool. The information included here is intended to provide guidance on the following topics:

- PL/SQL Service Consumer
- PL/SQL Provider Service
- Java EE Web Service Consumer
- Java EE Service Provider
- Web Service Call as a Remote EJB Call
- Web Service Call as a POJO Call
- Deploying the Web Service
- Creating a JDBC Data Source

Important Note About this Chapter

The implementation notes in this chapter are intended to provide some guidance in the development and deployment of the Web service layer. This information does not take into account the implementation of the business logic required to complete the application API layer.

The RSE tool and approaches described in this section are complex. A high level of skill and knowledge of the product is required to complete these implementation tasks. Also required is technology specific development of application APIs and the business logic that is needed to complete it.

Any issues that may arise with development tools, development environments, custom APIs, or custom message flows are the responsibility of the customer and not Oracle Retail.

PL/SQL Service Consumer Implementation Notes

To set up the Web service consumer side proxies, complete the following steps:

Note: See the section, "Important Note About this Chapter".

1. loadjava -u <username>/<password>@<host>:<port>:<SID> -r -v -f -genmissing dbwsclientws.jar dbwsclientdb102.jar

   Note: **loadjava is** a utility available in Oracle Database.

2. Edit and run *_grant.sql script as sysdba to give the user proper permission.
3. loadjava -u <username>/<password>@<host>:<port>:<SID> -r -v -f -genmissing *Consumer.jar.

   Note: If the jar already is loaded, drop the jar. If you get ORA-29533 while dropping the jar, drop the individual files.

   For example:

   ```
   dropjava -u
   <username>/<password>@<host>:<port>:<SID>packageName/SourceName
   ```

4. Run the *Consumer_create.sql in the schema that will use this API. The schema owner is user granted permission in Step 2.
5. Write a PL/SQL procedure to work as the client to call the Web service. A sample is provided below:

   Note: The following sample code is written for the PayTerm Web service. Replace the service endpoint URL and the consumer class name according to the Web service for which the client is generated.

```
create or replace PROCEDURE wstestClient IS
BEGIN
PayTermServiceConsumer.setEndpoint('http://10.141.26.93:7001/PayTermBean/PayTer
mService');
dbms_output.PUT_LINE(PayTermServiceConsumer.getEndPoint());
dbms_output.PUT_LINE(PayTermServiceConsumer.ping('TestMessage'));
dbms_output.PUT_LINE('Done.');
END;
```

PL/SQL Provider Service Implementation Notes

The distribution (.zip) file includes an .ear file that contains all the generated code for the service; it is ready to deploy to the application server. The business logic can be implemented in PL/SQL packages in Oracle. The distribution contains the "spec" and body scripts for the packages called by the deployed service.

To complete implementation, follow these steps:

Note: See the section, "Important Note About this Chapter".

1. Create the PL/SQL service provider distribution file using the RSE tool. The output of this process is the .zip file.

   Note: See Chapter 4,"User Interface Usage".

2. Extract the <service_name>.ProviderImplSpec.sql and <service_name>ProviderImplBody.sql files from the distribution zip file.
3. These files will be modified to provide a PL/SQL implementation for the service.
4. Extract the <service_name>-service.ear file from the distribution zip file. This file is the generated Web service that will be deployed.
5. Create the JDBC data source.

   Note: See "Creating a JDBC Data Source".

6. If not already deployed, deploy the Oracle Objects to the appropriate database user.

   Note: See the *Oracle Retail Functional Artifact Generator Guide*.

7. Modify the PL/SQL body file for the business logic implementation. The <service_name>ProviderImplBody.sql file contains comments about where to implement logic for each method on the service.
8. Install the modified PL/SQL packages to the database. They will be called by the Web service methods.
9. Deploy the <service_name>.ear file to the Oracle WebLogic Server.

Java EE Service Consumer Implementation Notes

The Java Web service consumer artifacts generated by this tool are based on the JAX-WS 2.1 specification. Services can be invoked in synchronous and asynchronous mode by using these artifacts.

To complete implementation, follow these steps:

Note: See the section, "Important Note About this Chapter".

1. Create a Web service client.
2. Create the application that uses the {WebsSrviceName}ServiceConsumer.jar and code the your Web service client. The {WebsSrviceName}ServiceConsumer.jar contains all necessary code to invoke the {WebsSrviceName}Service WebService.
3. Additional JAX-WS library jars might be required.
4. Deploy the service in the server.
5. Invoke the Web service client to see the results.

Sample Client Code

The code below is an example of how to invoke Oracle Retail's PayTerm Web service. For each Web service, a specific WebServiceConsumer code/jar must be generated that can "talk to" the service.

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

```
import java.math.BigDecimal;
import java.net.URL;
import javax.xml.namespace.QName;
import com.oracle.retail.integration.base.bo.paytermdesc.v1.PayTermDesc;
import com.oracle.retail.integration.base.bo.paytermref.v1.PayTermRef;
import
com.oracle.retail.rms.integration.services.paytermservice.v1.PayTermPortType;
import
com.oracle.retail.rms.integration.services.paytermservice.v1.PayTermService;
import junit.framework.TestCase;

public class PayTermTest extends TestCase{
   public void testCreatePayTerm(){
          try{
                  //qname is the namespace of the web service
                  QName qName = new
QName("http://www.oracle.com/retail/rms/integration/services/PayTermService/v1",
"PayTermService");

                  //wsdlLocation is the URL of the WSDL of the web service
                  URL wsdlLocation = new
URL("http://10.141.26.93:7001/PayTermBean/PayTermService?WSDL");
```

```
                  //get the web service instance
                  PayTermService service = new PayTermService(wsdlLocation,qName);
                  PayTermPortType port = service.getPayTermPort();

                  //populate input object for the web service method
                  PayTermDesc desc = new PayTermDesc();
                  desc.setTerms("terms");
                  desc.setDiscdays("1");
                  desc.setDueDays("1");
                  desc.setEnabledFlag("t");
                  desc.setPercent(new BigDecimal("1"));
                  desc.setRank("1");
                  desc.setTermsCode("code");
                  desc.setTermsDesc("desc");
                  desc.setTermsXrefKey("key");

                  //call the web service method. here ref is the response object
of the web service.
                  PayTermRef ref =  port.createPayTermDesc(desc);

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

Java EE Service Provider Implementation Notes

The RSE tool creates the appropriate provider Web service end-points as well as a skeleton implementation layer where the developer implements business logic. All of this is packaged inside the provider distribution archive file.

The Java EE Provider distribution file provides a sample deployable application and all the libraries that can be used to create Web services using retail payloads. The distribution file follows the naming convention of <*appname*>_JavaEEServiceProvider.zip. For example, the distribution file for the RMS application is named rms_JavaEEServiceProvider.zip. The <rms> prefix must be replaced with the name of any other application being developed.

The Web services generated by the RSE tool can be implemented and deployed in a number of ways. This section includes three implementation use cases for reference.

Note: See the section, "Important Note About this Chapter".

Use Case 1: Complete the Generator Provided Stub Code Implementation

1. Generate the distribution file using the RSE tool.
2. Extract the <service_name>-ejb-impl-src.jar file from the zip file.
3. Extract the <service_name>-service.ear file from the zip file.
4. Add business logic code where indicated in the Impl java files.
5. Use the java jar command to re-build the <service_name>-service-ejb-impl.jar file.
6. Use the jar command to update .ear file with the new implementation jar.
7. Deploy the .ear file to the server.

Use Case 2: Provide a Custom impl jar to the RSE Tool

1. Create custom java classes that implement the <service_name>ServiceProvider interfaces contained in the <service_name>-service-ejb.jar file.
2. Extract the ServiceProviderImplLookupFactory.properties file from the .ear file.
3. Modify the properties file to point to your implementation classes for the services.
4. Use the jar command to create a jar containing your implementation classes, as well as the modified properties file.
5. Run the RSE tool again and provide the new custom implementation jar file.
6. Extract and deploy the generated .ear file to the server.

Use Case 3: Package the Generated Service Classes in an Existing Application

1. Generate the distribution file using the RSE tool.
2. The service interfaces are provided in the <*appname*>-service-ejb.jar file in the distribution file. This jar file should be included in the application classpath.
3. Source code of sample implementations for the service interfaces are provided in the <*appname*>-service-ejb-src.jar file in the distribution file. (If application developers want to use the same classes in their application, they can extract the java files from the jar file and include those in application source code. They also can add their own business logic in the method implementations. If they decide to write their own implementations, they should make sure that the appropriate service interfaces are implemented.)
4. After writing the Web service implementations, the java files should be compiled. The class files can be included in a new jar file or in the same jar file used for the rest of the classes of the application.
5. Modify the ServiceProviderImplLookupFactory.properties file to include appropriate class names of service implementations and include it in application classpath. A recommended approach is to include the properties file in the jar file that contains the service implementation classes.
6. Make sure that the following jar files are included in the application ear file:
   - <*appname*>-service-ejb.jar
   - Jar file containing the service implementation classes
   - jaxb-api.jar
   - retail-public-payload-java-beans-base.jar
   - retail-public-payload-java-beans.jar
   - retail-soa-enabler.jar
7. Include an ejb-module in the application.xml of the application. The module name should be same as the name of <*appname*>-service-ejb.jar file.
8. The .ear file is ready for deployment on the server.

Web Service Call as a Remote EJB Call

This section applies to PL/SQL Web service implementations and Java EE Web service implementations.

A client can call a Web service as a remote EJB call to improve performance by avoiding marshalling and unmarshalling.

Note: See the section, "Important Note About this Chapter".

Prerequisites

The following is a list of prerequisites to implementation.

1. Get the updated wlfullclient.jar (integration-lib\third-party\oracle\wl\10.3\)& retail-soa-enabler.jar (integration-lib\internal-build\rse\) from the Repository.
2. Run build.xml for retail-soa-enabler.
3. Generate the .ear and deploy it to server.
4. Configure the data source in the server.

Procedure

Complete the following steps.

1. Create a Java file containing the code below inside any package. (See code sample at the end of this section.)
2. Include the following jar files in the classpath:
   - retail-public-payload-java-beans-base.jar
   - retail-public-payload-java-beans.jar
   - oo-jaxb-bo-converter.jar
   - retail-soa-enabler.jar
   - <*appname*>-service-ejb.jar
3. Run code as a Java application.

Note: The sample code below obtains a context for accessing the WebLogic naming service and calls a lookup method to get the Object inside the container by providing a binding name. It then calls a corresponding Web service method. As an example, the code sample calls the PayTerm service.

```
import java.util.Properties;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import com.oracle.retail.integration.base.bo.paytermdesc.v1.PayTermDesc;
import com.oracle.retail.integration.base.bo.paytermref.v1.PayTermRef;
import
com.oracle.retail.integration.services.exception.v1.EntityNotFoundWSFaultException
;
import
com.oracle.retail.integration.services.exception.v1.IllegalArgumentWSFaultExceptio
```

```
n;
import
com.oracle.retail.integration.services.exception.v1.IllegalStateWSFaultException;
import com.oracle.retail.rms.integration.services.paytermservice.v1.PayTermRemote;


public class WebLogicEjbClient {

        public static void main(String[] args) throws NamingException,
IllegalArgumentWSFaultException, EntityNotFoundWSFaultException,
IllegalStateWSFaultException {

               Context ctx = getInitialContext("t3://localhost:7001",
"weblogic","weblogic");
Object ref = ctx .lookup("PayTerm#com.oracle.retail.rms.integration.services.
paytermservice.v1.PayTermRemote");

               PayTermRemote remote = (PayTermRemote)(ref);

               PayTermRef ref = new PayTermRef();
               PayTermDesc desc = remote.findPayTermDesc(ref);

               System.out.println("findPayTermDesc=" + desc);

}

static Context getInitialContext(String url, String user, String password)
throws NamingException {

      Properties h = new Properties();
      h.put(Context.INITIAL_CONTEXT_FACTORY,
      "weblogic.jndi.WLInitialContextFactory");
      h.put(Context.PROVIDER_URL, url);
      h.put(Context.SECURITY_PRINCIPAL, user);
      h.put(Context.SECURITY_CREDENTIALS, password);
      return new InitialContext(h);

     }
}
```

Code Description

Code sample 1:

```
Context ctx = getInitialContext("t3://localhost:7001", "weblogic","weblogic");
```

Description: Gets Initial Context object by passing the URL (local WebLogic URL, if not configured to other), user name, and password of the server.

Code sample 2:

```
Object ref = ctx .lookup("PayTerm#com.oracle.retail.rms.integration.services.
paytermservice.v1.PayTermRemote");
```

Description: Lookup method retrieves the name of Object. Throws naming exception if the binding name is missing from the server. Binding name can be found after deploying the .ear file to the server, at JNDI Tree Page. (Summary of Servers >examplesServer>view JNDI Tree).

Code sample 3:

```
PayTermRemote remote = (PayTermRemote)(ref);
```

Description: Create PayTermRemote object by casting ref object.

Code sample 4:

```
PayTermRef ref = new PayTermRef();
PayTermDesc desc = remote.findPayTermDesc(ref);
```

Description: Invoked Web service method findPayTermDesc as a remote call. Depending on the requirement, the user can vary the binding name and create a different object to invoke the Web service deployed to the server as a remote EJB call using the above code.

Web Service Call as a POJO Call

This section applies to PL/SQL Web service implementations and Java EE Web service implementations.

If an application is a core Java application, it can still call the Web services classes, but as POJO classes. In this case, the Web service classes act as simple Java classes, and there is no marshalling of XML involved, nor a remote call as an EJB.

The PL/SQL provider services need a database connection to call PL/SQL packages. In the case of a Web service call or an EJB call, the service gets the connection from the data source supplied by the Java EE container through resource injection. But in the case of a Java application, the data source is not available through this mechanism. The connection must be passed to the Web service class before invoking any business methods on it. To achieve this, the caller application must create an instance of the Web service class using the non-default constructor available in the service bean class. An example of the signature of the constructor is below:

```
public PayTermBean(Connection conn,Map<String,String> serviceContext)
```

Note: The bean class is available in the <*appname*>-service-ejb.jar for each Web service generated. For example, if the service name is PayTerm in the service definition XML, the name of the generated bean class will be PayTermBean. This is the class that should be used to call a Web service as a POJO.

In the constructor shown above, the first parameter is for database connection. The second parameter is for the calling application to provide any additional parameters to the bean passed on to the PL/SQL package. When the bean is called as a Web service, an instance of ServiceOpContext class is created by using properties available from an instance of javax.xml.ws.WebServiceContext, available through resource injection. When the bean is called as EJB, then an instance of ServiceOpContext is created from the values in an instance of javax.ejb.EJBContext, available through resource injection. But when the bean is called as a POJO, none of these objects is available. Therefore, a map has been added in the constructor so that the calling application can set the required values. If a null object is passed to the constructor for the map, an empty instance of ServiceOpContext is created. If the map contains a key named "user," a Principal object is created with the value of that key, and it is set in the ServiceOpContext object.

Procedure

Complete the following steps.

Note: See the section, "Important Note About this Chapter".

1. Generate the .ear file for Web services and extract the following jar files from it:
   - retail-public-payload-java-beans-base.jar
   - retail-public-payload-java-beans.jar
   - oo-jaxb-bo-converter.jar
   - retail-soa-enabler.jar
   - <*appname*>-service-ejb.jar
2. Include these jar files in the classpath of the Java application that is going to invoke the beans as POJO classes.
3. Write the code to call the bean classes. (Sample code is provided below in this section.)
4. Run the calling class.

Note: The connection must be committed or rolled back by the calling application. Because there is no Java EE container available in this case, the bean cannot start and end a transaction. Therefore, it is the responsibility of the calling application to manage the transaction and the connection. In the following sample code, the calling class is committing the connection in case of a successful response from the bean, and it is rolling back the connection in case of any exception thrown by the Web service. The calling application determines how it wants to handle exceptions.

Sample Code for POJO Invocation

```
public class PayTermService extends TestCase{

       public void testPayTerm(){
               Connection conn = null;
               try{
                     //get the database connection
                     Class.forName("oracle.jdbc.OracleDriver");
                     conn
=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","stubby","ret
ek");

                    //create map for ServiceOpContext
                    Map<String,String> ctxMap = new HashMap<String, String>();
                    ctxMap.put("user", "user1");

                    //instantiate the web service bean class
                    PayTermBean bean = new PayTermBean(conn,ctxMap);

                    //populate the input object for web service method
                    PayTermRef ref = new PayTermRef();
                    ref.setTerms("terms");
                    ref.setTermsXrefKey("key");

                    //call the web service.here desc is the response object
                    PayTermDesc desc = bean.findPayTermDesc(ref);

                    //print the response object value
                    System.out.println("desc value="+desc.getTerms());

                    //commit the database connection
                    conn.commit();
            }catch(Exception e){
                     e.printStackTrace();
                     try{
                            conn.rollback();
                     }catch(SQLException se){
                             se.printStackTrace();
                     }
            }finally{
                     if(conn !=null){
                             try{
                                    conn.close();
                            }catch(SQLException se){
                                    se.printStackTrace();
                            }
                      }
              }
      }
}
```

Deploying the Web Service

This section applies to PL/SQL Web service implementations and Java EE Web service implementations.

> **Note:** See the section, "Important Note About this Chapter".

Complete the following steps using the WebLogic Server Administration Console:

1. Navigate to the Deployments page.
2. Click **Install**.

> **Note:** If the service application has already been installed, see "Redeploy the Service Application".

3. The **Locate deployment to install and prepare for deployment** screen is displayed. Follow the instructions to locate the <service-name>.ear file on the WebLogic Server host

   If rib-home is located on a host other than the Oracle WebLogic Server, select Upload Files. On the **Upload a Deployment to the admin server** screen, use the browse button to locate the <service-name>.ear file in the Deployment Archive.
4. Select the igs-service.ear.
5. Click **Next** to move to **Choose targeting style**.
6. Select **Install this deployment as an application**.

7. Click **Next** to move to **Optional Settings**.
8. Click **Next** to move to **Review your choices and click Finish**.
9. Select **No, I will review the configuration later**.
10. Click **Finish** to deploy the application.

Redeploy the Service Application

If the <*service-name*> application has already been deployed, follow these steps:

1. If the <*service-name*> application is running, select **Stop** and **When Work Completes** or **Force Stop Now**, depending on the environment. The recommended option always is **When Work Completes**.
2. Select **Delete**.
3. The Summary of Deployments should now include the igs-service.
4. Return to "Deploying the Web Service".

Verify the Service Application Installation Using the Administration Console

To verify the Service installations using the Oracle WebLogic Administration Console, follow these steps.

Note: See Oracle WebLogic Server 11g Release 3 (10.3.3) documentation about the Administration console.

1. Navigate to the Deployments screen.
2. Locate the <service-name> on the Summary of Deployments screen.
3. Click **plus sign** next to the ig-service to expand the tree.
4. Locate the Web services section.
5. Click any Web service to move to a Settings for <service name> Service screen.
6. Click the **Testing** tab.
7. Click **plus sign** next to the service name to expand the tree.
8. Click the **Test Clien**t link to move to the WebLogic Test Client screen.
9. Select **Ping Operation**.
10. The test page will show the request message and the response message.

Creating a JDBC Data Source

This section applies to PL/SQL Web service implementations and to Java EE Web service implementations.

To create a JDBC Data Source, follow these steps:

Note: See the section, "Important Note About this Chapter".

1. Log in to the WebLogic administration console. Use the URL, http://<host>:<listen port>/console/login/LoginForm.jsp.
2. Navigate the domain structure tree to Services/JDBC/Data Sources.
3. Click **New** to start creating the new Data Source. Enter the required information:

   Name: Enter any name for the data source.

   JNDI name: This field must be set to jdbc/RetailWebServiceDs. The generated code for the service will use this JNDI name to look up the data source.
4. Select the transaction options for your data source and click **Next.**
5. Enter the database name and user information for the data source. Click **Next**.
6. The screen includes the connection information for your data source. Click **Test Configuration** to ensure the connection information is correct. If it is correct, the following message is displayed: "Connect test succeeded."
7. Click **Next** and select a server to deploy the data source to. This is not necessary at this point if you want to deploy the data source to a server at a later time.
8. Click **Finish** to complete the data source setup. The new data source is displayed on the data sources screen.
9. Click the new data source to view the properties. A default connection pool is created for the data source. Click the **Connection Pool** tab to view the connection pool properties.
10. The generated JDBC connection URL for the data source is displayed. The Oracle URL is formatted as follows: jdbc:oracle:thin:@<hostname>:<port>:<sid>.

    For example: jdbc:oracle:thin:@localhost:1521:orc
11. If the database is a RAC database, the URL should be in the following format

    ```
    jdbc:oracle:thin:@(DESCRIPTION =(ADDRESS_LIST =(ADDRESS = (PROTOCOL = TCP)(HOST
    = <host>)(PORT = <port>))(ADDRESS = (PROTOCOL = TCP)(HOST = <host>)(PORT =
    <port>))(LOAD_BALANCE = yes))(CONNECT_DATA =(SERVICE_NAME = <sid>)))
    ```

    For example:

    ```
    jdbc:oracle:thin:@(DESCRIPTION =(ADDRESS_LIST =(ADDRESS = (PROTOCOL = TCP)(HOST
    = mspvip72)(PORT = 1521))(ADDRESS = (PROTOCOL = TCP)(HOST = mspvip73)(PORT =
    1521))(LOAD_BALANCE = yes))(CONNECT_DATA =(SERVICE_NAME = dvolr02)))
    ```
12. Restart the WebLogic instance to apply the data source changes.

9 Web Services Security Setup Guidelines

Web services can be secured in many ways. The ws-policy tab of Web service in the WebLogic admin console lists the various policy files that can be used to secure Web services. This chapter describes the following methods for securing Web services.

- Simple user name and password authentication
- Password encryption using certificates

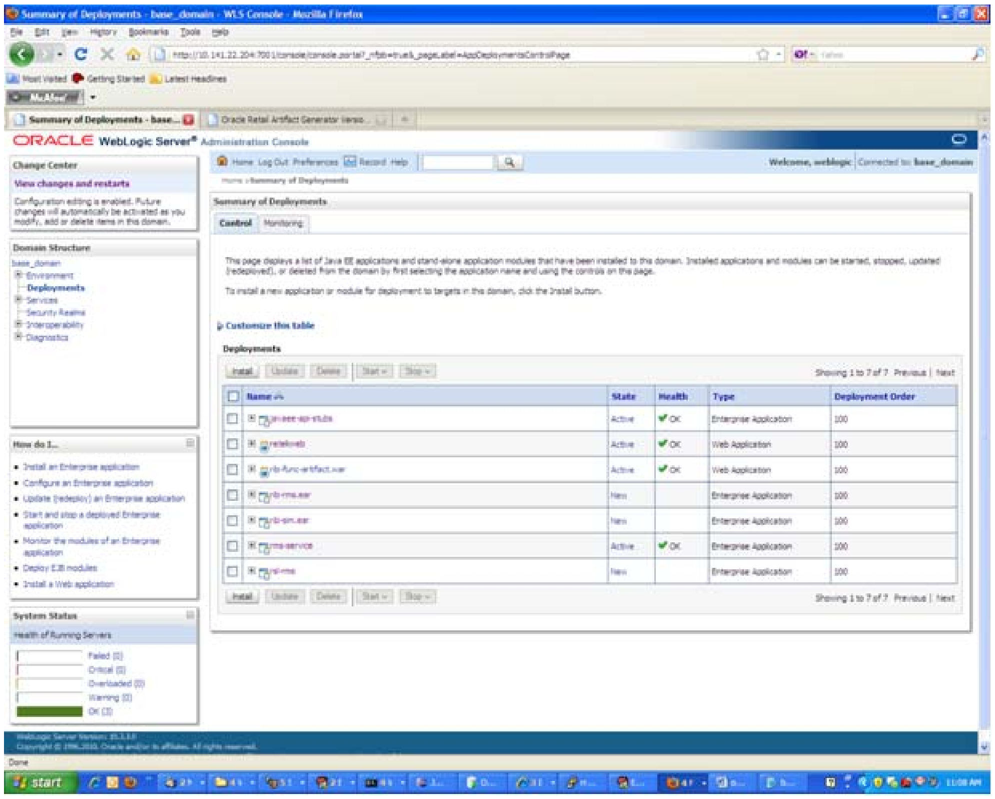
Server-Side Setup

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

Attach 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 the application. In the illustration below, the application is rms-service.



Summary of Deployments

Change Center

View changes and restarts

Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

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

Customize this table

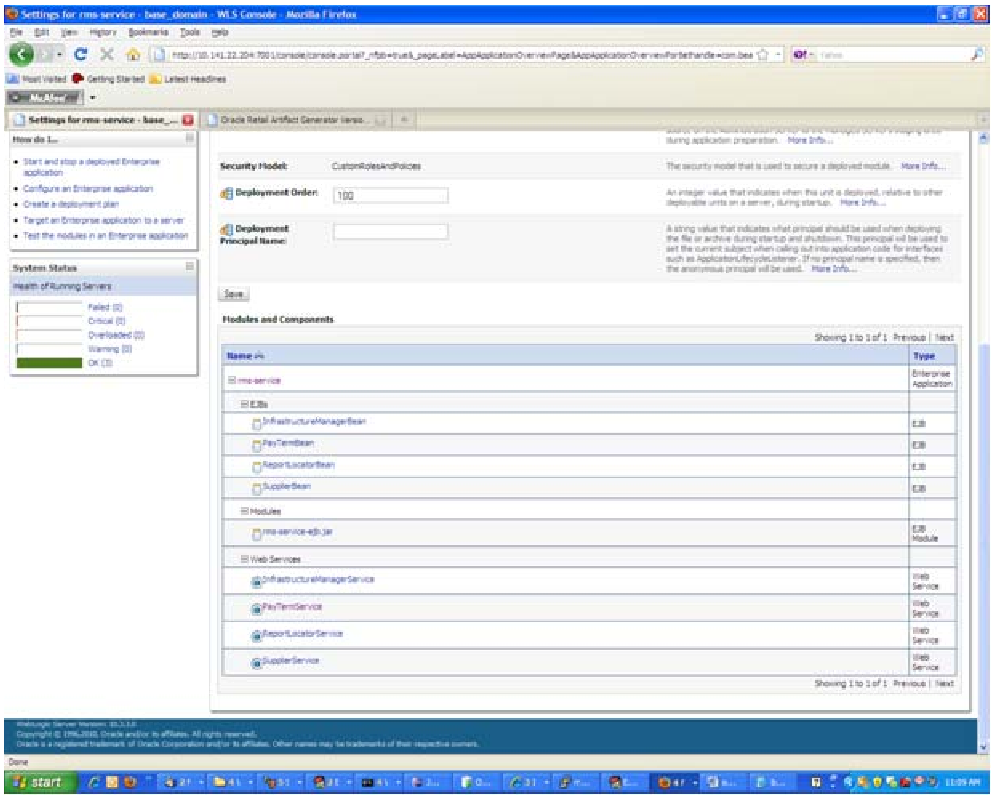
Deployments

Showing 1 to 7 of 7 Previous | Next

Name	State	Health	Type	Deployment Order
javaee-api-stubs	Active	✓ OK	Enterprise Application	100
retailweb	Active	✓ OK	Web Application	100
rib-func-artifact.war	Active	✓ OK	Web Application	100
rib-rms.ear	New		Enterprise Application	100
rib-sim.ear	New		Enterprise Application	100
rms-service	Active	✓ OK	Enterprise Application	100
rsl-rms	New		Enterprise Application	100

Showing 1 to 7 of 7 Previous | Next

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



Settings for rms-service - base_domain - WLS Console - Mozilla Firefox

How do I...

- Start and stop a deployed Enterprise application
- Configure an Enterprise application
- Create a deployment plan
- Target an Enterprise application to a server
- Test the modules in an Enterprise application

System Status

Health of Running Servers

Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
OK (3)

Security Model:	CustomRolesAndPolicies	The security model that is used to secure a deployed module. More Info...
Deployment Order:	100	An integer value that indicates when this unit is deployed, relative to other deployable units on a server, during startup. More Info...
Deployment Principal Name:		A string value that indicates what principal should be used when deploying the file or archive during startup and shutdown. This principal will be used to set the current subject when calling out into application code for interfaces such as ApplicationLifecycleListener. If no principal name is specified, then the anonymous principal will be used. More Info...

Save

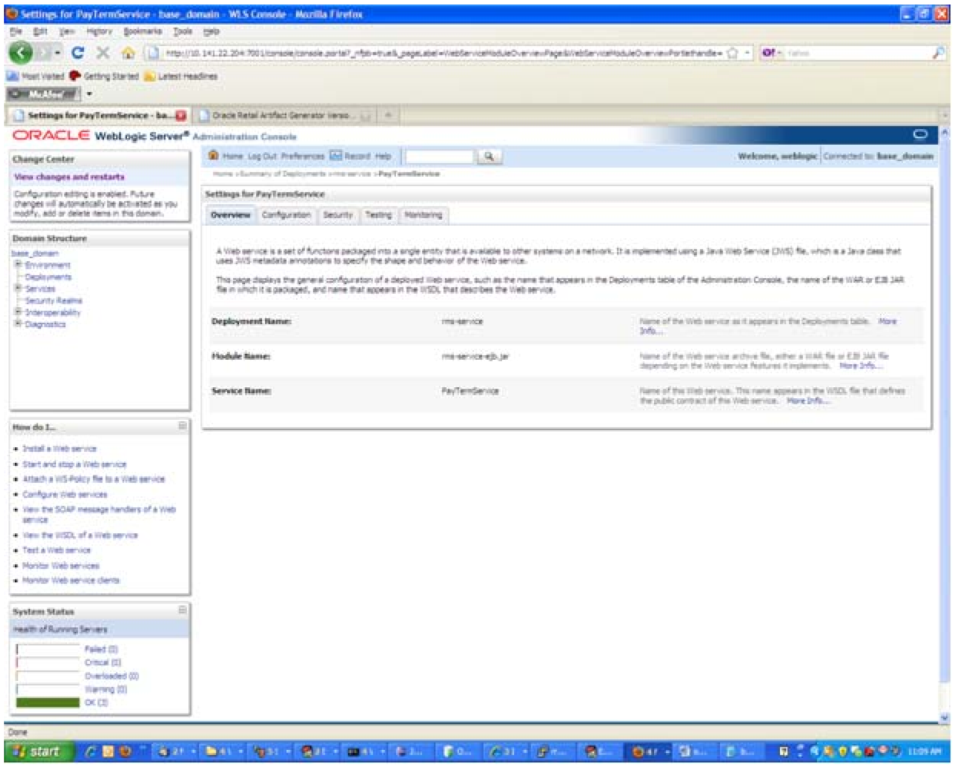
Modules and Components

Showing 1 to 1 of 1 Previous | Next

Name	Type
rms-service	Enterprise Application
EJBs	
InfrastructureManagerBean	EJB
PayTermBean	EJB
ReportLocatorBean	EJB
SupplierBean	EJB
Modules	
rms-service-ejb.jar	EJB Module
Web Services	
InfrastructureManagerService	Web Service
PayTermService	Web Service
ReportLocatorService	Web Service
SupplierService	Web Service

Showing 1 to 1 of 1 Previous | Next

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


Settings for PayTermService - base_domain - WLS Console - Mozilla Firefox
ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Change Center
View changes and restarts
Domain Structure
How do I...
System Status
Overview Configuration Security Testing Monitoring

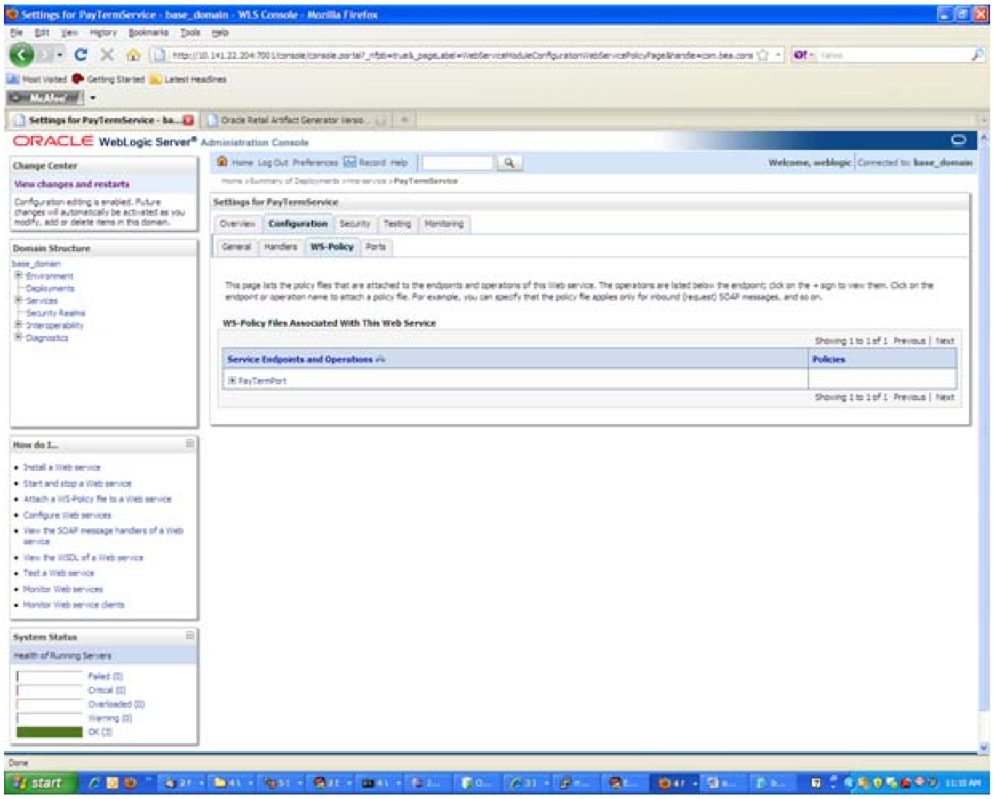
Settings for PayTermService

A Web service is a set of functions packaged into a single entity that is available to other systems on a network. It is implemented using a Java Web Service (JWS) file, which is a Java class that uses JWS metadata annotations to specify the shape and behavior of the Web service.

This page displays the general configuration of a deployed Web service, such as the name that appears in the Deployments table of the Administration Console, the name of the WAR or EJB JAR file in which it is packaged, and name that appears in the WSDL that describes the Web service.

Deployment Name:	rms-service	Name of the Web service as it appears in the Deployments table. More Info...
Module Name:	rms-service-ejb.jar	Name of the Web service archive file, either a WAR file or EJB JAR file depending on the Web service features it implements. More Info...
Service Name:	PayTermService	Name of this Web service. This name appears in the WSDL file that defines the public contract of this Web service. More Info...

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


Settings for PayTermService - base_domain - WLS Console - Mozilla Firefox
ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Overview Configuration Security Testing Monitoring
General Handlers WS-Policy Ports

Settings for PayTermService

This page lists the policy files that are attached to the endpoints and operations of this Web service. The operations are listed below the endpoint; click on the + sign to view them. Click on the endpoint or operation name to attach a policy file. For example, you can specify that the policy file applies only for inbound (request) SOAP messages, and so on.

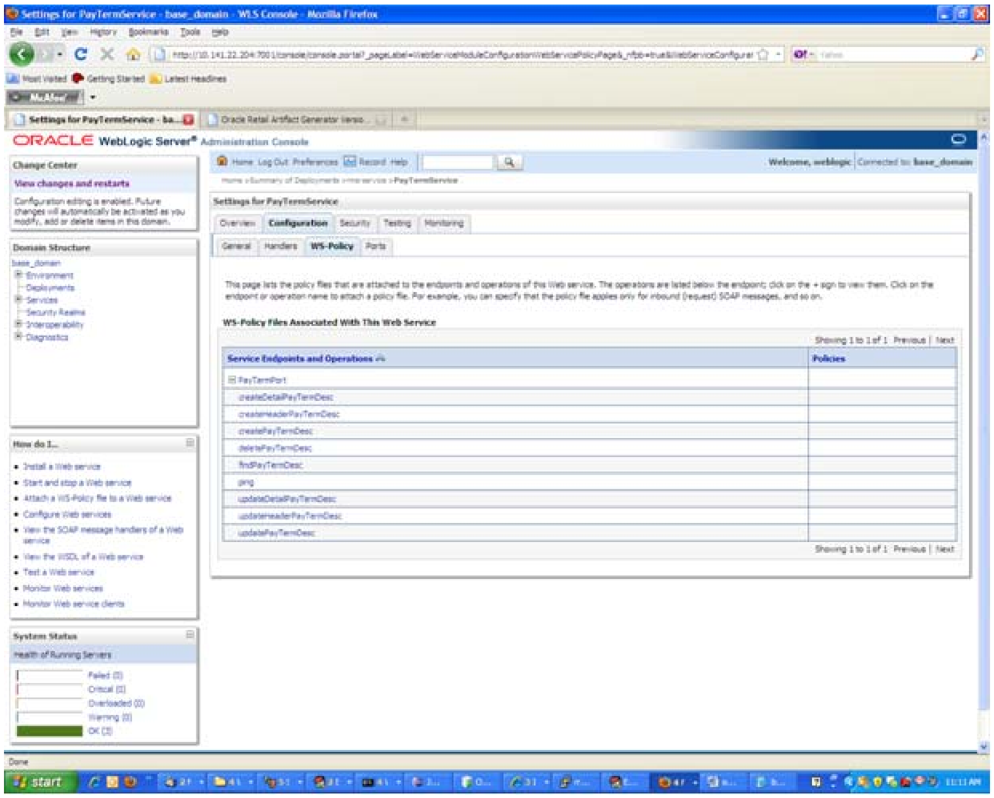
WS-Policy Files Associated With This Web Service

Showing 1 to 1 of 1 Previous | Next

Service Endpoints and Operations	Policies
⊞ PayTermPort	

Showing 1 to 1 of 1 Previous | Next

5. Click the **+** next to the port name. The Web service operations are displayed:



Settings for PayTermService

This page lists the policy files that are attached to the endpoints and operations of this Web service. The operations are listed below the endpoint; click on the + sign to view them. Click on the endpoint or operation name to attach a policy file. For example, you can specify that the policy file applies only for inbound (request) SOAP messages, and so on.

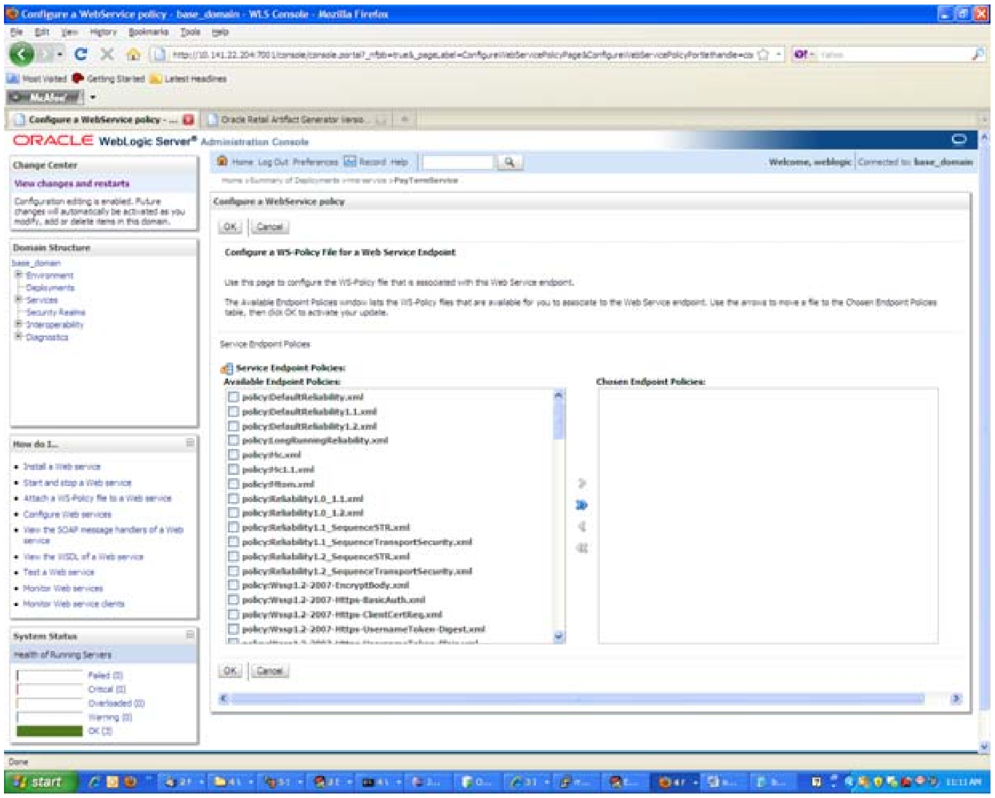
WS-Policy Files Associated With This Web Service

Showing 1 to 1 of 1 Previous | Next

Service Endpoints and Operations	Policies
PayTermPort	
createDetailPayTermDesc	
createHeaderPayTermDesc	
createPayTermDesc	
deletePayTermDesc	
findPayTermDesc	
ping	
updateDetailPayTermDesc	
updateHeaderPayTermDesc	
updatePayTermDesc	

Showing 1 to 1 of 1 Previous | Next

6. You can secure all the Web service operations 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:



Configure a WebService policy

Configure a WS-Policy File for a Web Service Endpoint

Use this page to configure the WS-Policy file that is associated with this Web Service endpoint.

The Available Endpoint Policies window lists the WS-Policy files that are available for you to associate to the Web Service endpoint. Use the arrows to move a file to the Chosen Endpoint Policies table, then click OK to activate your update.

Service Endpoint Policies

Service Endpoint Policies:

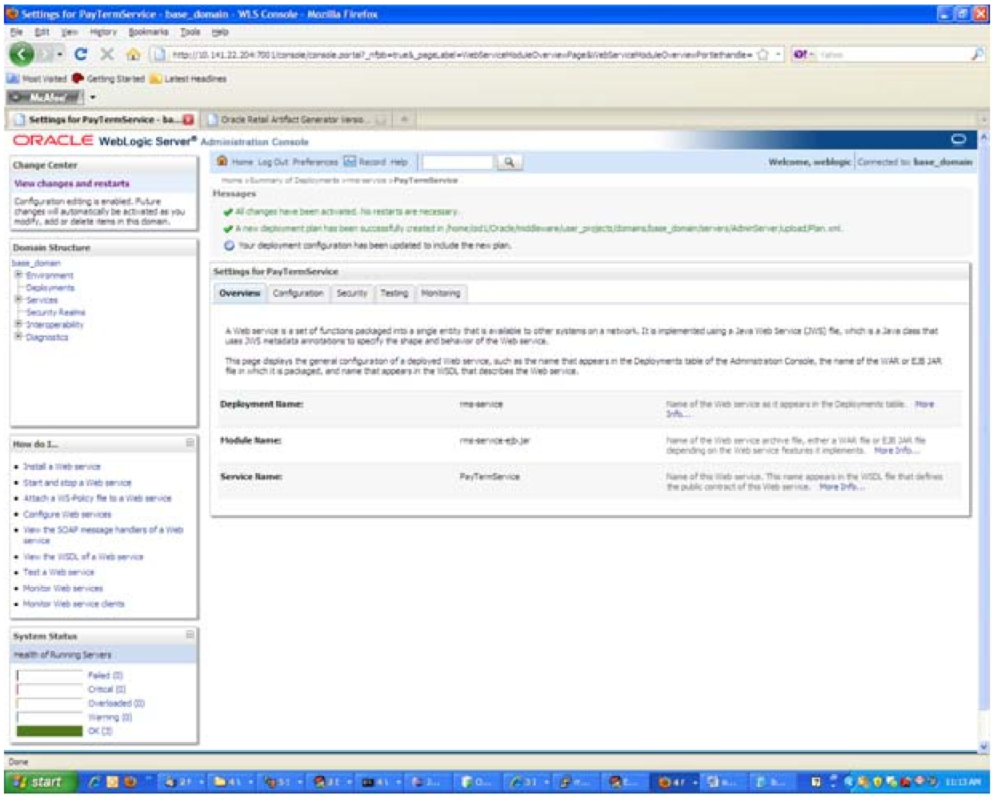
Available Endpoint Policies:

- policy:DefaultReliability.xml
- policy:DefaultReliability1.1.xml
- policy:DefaultReliability1.2.xml
- policy:LongRunningReliability.xml
- policy:Mc.xml
- policy:Mc1.1.xml
- policy:Mtom.xml
- policy:Reliability1.0_1.1.xml
- policy:Reliability1.0_1.2.xml
- policy:Reliability1.1_SequenceSTR.xml
- policy:Reliability1.1_SequenceTransportSecurity.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

Chosen Endpoint Policies:

OK
Cancel
Change Center
View changes and restarts
Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.
Domain Structure
base_domain
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
How do I...
Install a Web service
Start and stop a Web service
Attach a WS-Policy file to a Web service
Configure Web services
View the SOAP message handlers of a Web service
View the WSDL of a Web service
Test a Web service
Monitor Web services
Monitor Web service clients
System Status
Health of Running Servers
Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
OK (3)

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 **OK**. The Save Deployment Plan Assistant screen is displayed.
8. At the bottom of the Save Deployment Plan Assistant screen, click **OK**. The following screen is displayed, including status messages near the top:



Settings for PayTermService

Messages

All changes have been activated. No restarts are necessary.

A new deployment plan has been successfully created in /home/jsd1/Oracle/Middleware/user_projects/domains/base_domain/servers/AdminServer/upload/Plan.xml.

Your deployment configuration has been updated to include the new plan.

Overview | Configuration | Security | Testing | Monitoring

A Web service is a set of functions packaged into a single entity that is available to other systems on a network. It is implemented using a Java Web Service (JWS) file, which is a Java class that uses JWS metadata annotations to specify the shape and behavior of the Web service.

This page displays the general configuration of a deployed Web service, such as the name that appears in the Deployments table of the Administration Console, the name of the WAR or EJB JAR file in which it is packaged, and name that appears in the WSDL that describes the Web service.

Deployment Name:	rms-service	Name of the Web service as it appears in the Deployments table. More Info...
Module Name:	rms-service-ejb.jar	Name of the Web service archive file, either a WAR file or EJB JAR file depending on the Web service features it implements. More Info...
Service Name:	PayTermService	Name of this Web service. This name appears in the WSDL file that defines the public contract of this Web service. More Info...

ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Change Center
View changes and restarts
Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.
Domain Structure
base_domain
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
How do I...
Install a Web service
Start and stop a Web service
Attach a WS-Policy file to a Web service
Configure Web services
View the SOAP message handlers of a Web service
View the WSDL of a Web service
Test a Web service
Monitor Web services
Monitor Web service clients
System Status
Health of Running Servers
Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
OK (3)

9. On the Web Service page, under the Testing tab, click the WSDL to view the details of the policy just added to the Web service. The WSDL will contain information similar to the following:

```
<?xml version='1.0' encoding='UTF-8'?>
 <definitions
xmlns:wssutil="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecuri
ty-utility-1.0.xsd" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.oracle.com/retail/rms/integration/services/PayTermService
/v1" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.oracle.com/retail/rms/integration/services/PayTermS
ervice/v1" name="PayTermService">
<wsp:UsingPolicy wssutil:Required="true" />
<wsp:Policy wssutil:Id="usernametoken">
<ns1:SupportingTokens
xmlns:ns1="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512">
<wsp:Policy>
<ns1:UsernameToken
ns1:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/Inc
ludeToken/AlwaysToRecipient">
<wsp:Policy>
<ns1:WssUsernameToken10 />
</wsp:Policy>
</ns1:UsernameToken>
</wsp:Policy>
</ns1:SupportingTokens>
</wsp:Policy>
```

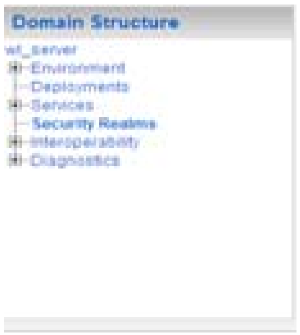
Create Roles and Users

This section describes how to add roles and users who can access the Web services.

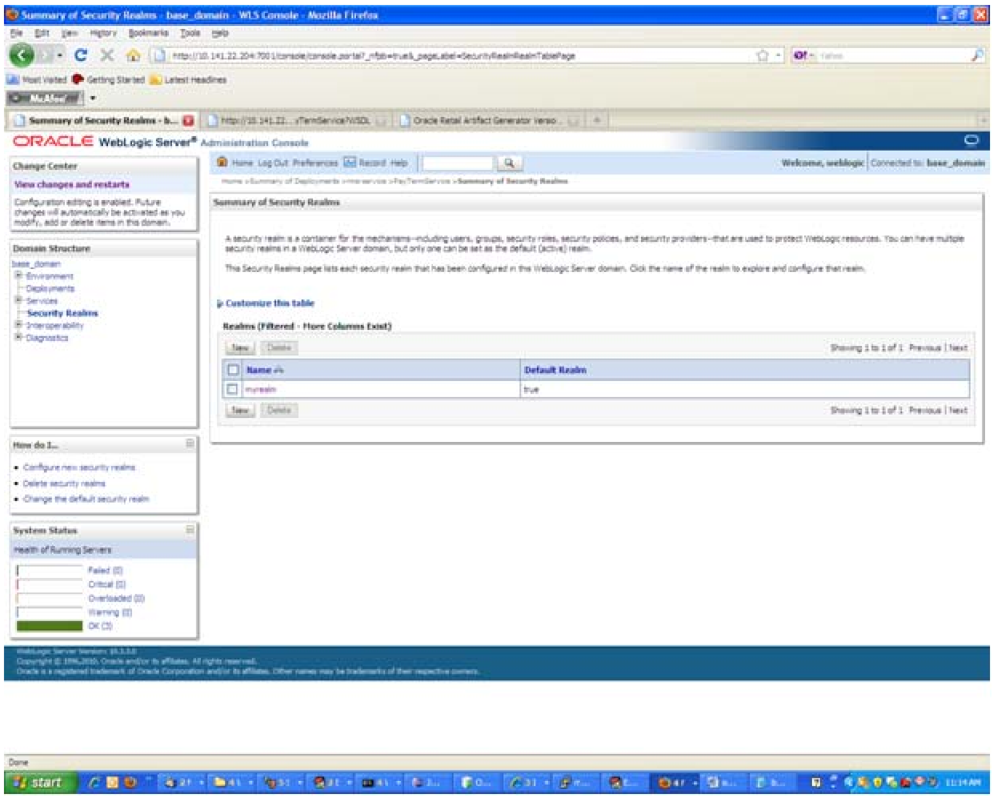
Note: To associate roless and users with Web services, make sure the .ear file containing the Web services has been deployed with the option, Custom Roles and Policies. This option is available from the Optional Settings page during the deployment of the .ear file.

---

1. Within the Oracle WebLogic Services Administration Console, click the **Security Realms** link in the Domain Structure window:



2. The Summary of Security Realms screen is displayed, including the name of the default realm:


Summary of Security Realms - base_domain - WLS Console - Mozilla Firefox
ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic | Connected to: base_domain
Change Center
View changes and restarts
Domain Structure
base_domain
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
How do I...
Configure new security realms
Delete security realms
Change the default security realm
System Status
Health of Running Servers
Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
OK (3)
Customize this table
New
Delete
Showing 1 to 1 of 1 Previous | Next

Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

Summary of Security Realms

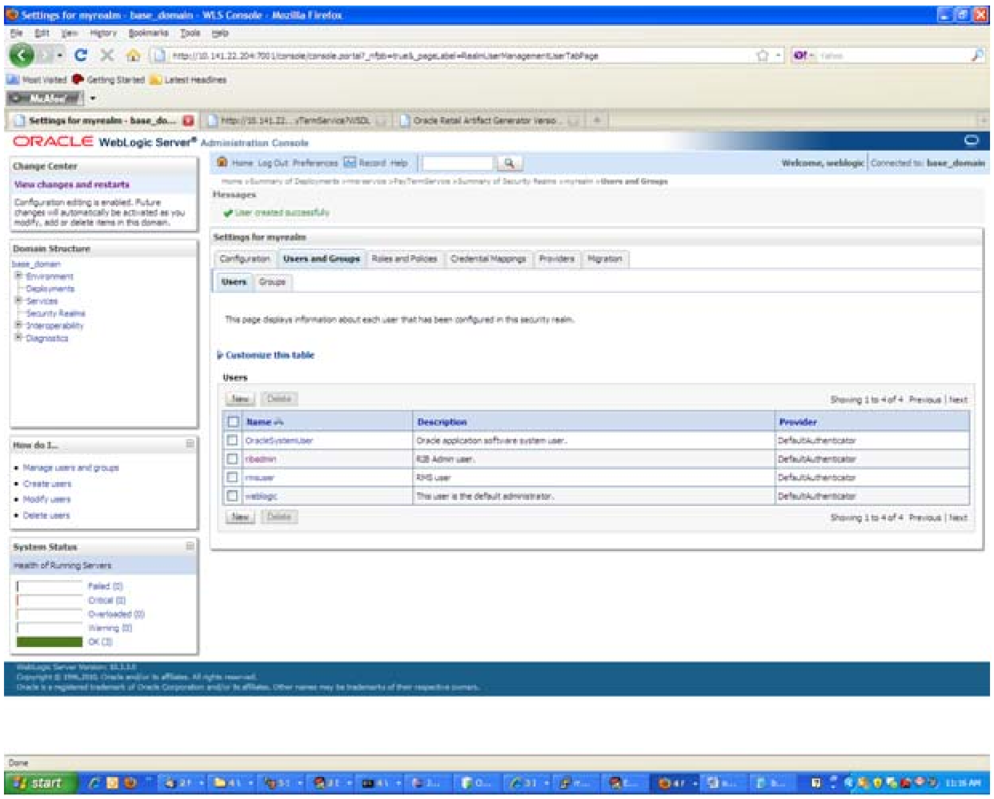
A security realm is a container for the mechanisms--including users, groups, security roles, security policies, and security providers--that are used to protect WebLogic resources. You can have multiple security realms in a WebLogic Server domain, but only one can be set as the default (active) realm.

This Security Realms page lists each security realm that has been configured in this WebLogic Server domain. Click the name of the realm to explore and configure that realm.

Realms (Filtered - More Columns Exist)

Name	Default Realm
myrealm	true

3. Click the name of the default realm. The settings for the realm are displayed.
4. On the Settings screen, click the **Users and Groups** tab.
5. 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.
6. In the Create a New User screen, enter a user name/password. Leave the default value for Provider. Click **OK** to save the information. The new user is added to the list of users.


Settings for myrealm - base_domain - WLS Console - Mozilla Firefox
ORACLE WebLogic Server Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Change Center
View changes and restarts
Domain Structure
How do I...
System Status
Configuration
Users and Groups
Roles and Policies
Credential Mappings
Providers
Migration
Users
Groups
Customize this table
New
Delete
Showing 1 to 4 of 4 Previous | Next

Messages

User created successfully

Settings for myrealm

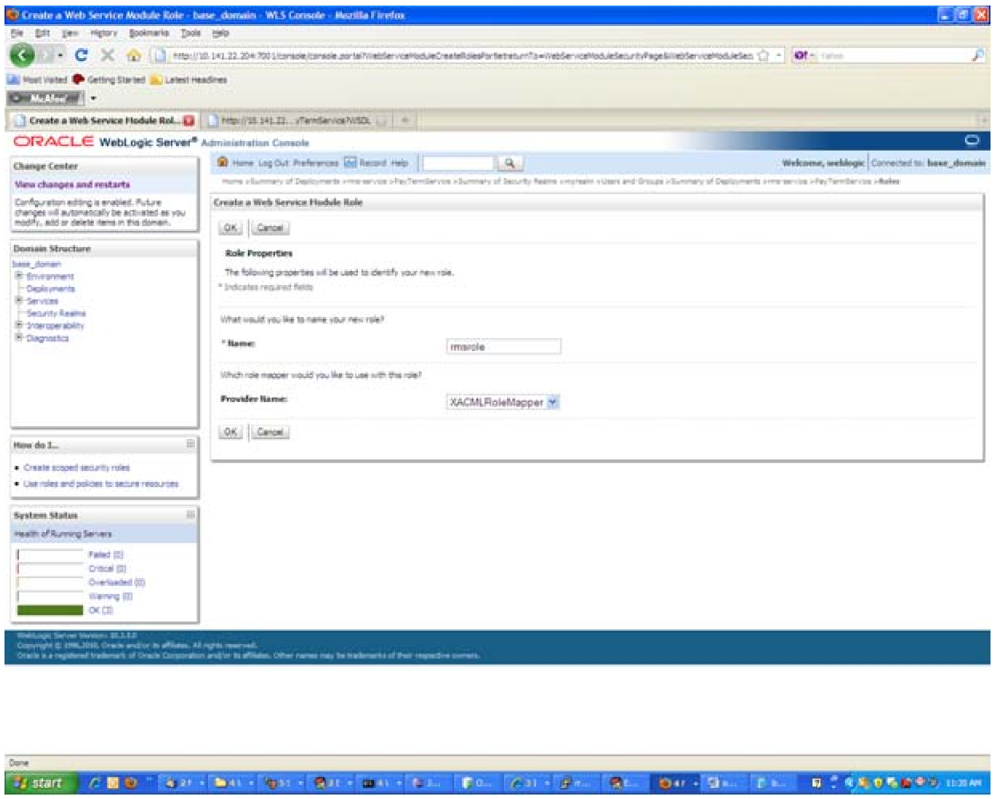
This page displays information about each user that has been configured in this security realm.

Users

Name	Description	Provider
OracleSystemUser	Oracle application software system user.	DefaultAuthenticator
rbadmin	RB Admin user.	DefaultAuthenticator
rmsuser	RMS user	DefaultAuthenticator
weblogic	This user is the default administrator.	DefaultAuthenticator

Note: Adding roles can be done 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.

7. Navigate to the Security tab of the Web service. Click the **Roles** tab.
8. In the Roles tab, click **New**. The Create a Web Service Module Role screen is displayed.



Create a Web Service Module Role

Role Properties

The following properties will be used to identify your new role.

* Indicates required fields

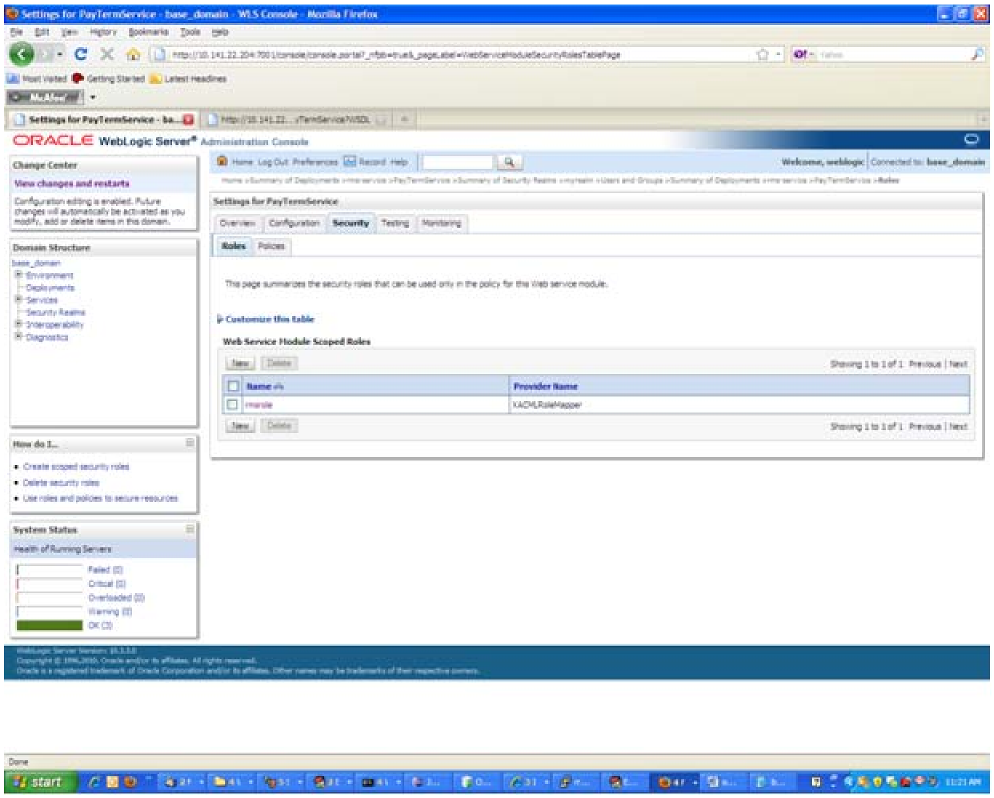
What would you like to name your new role?

* Name: rmsrole

Which role mapper would you like to use with this role?

Provider Name: XACMLRoleMapper

9. 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 Role tab of the Web service.



Settings for PayTermService - base_domain - WLS Console - Mozilla Firefox

Settings for PayTermService

Overview | Configuration | Security | Testing | Monitoring

Roles | Policies

This page summarizes the security roles that can be used only in the policy for this Web service module.

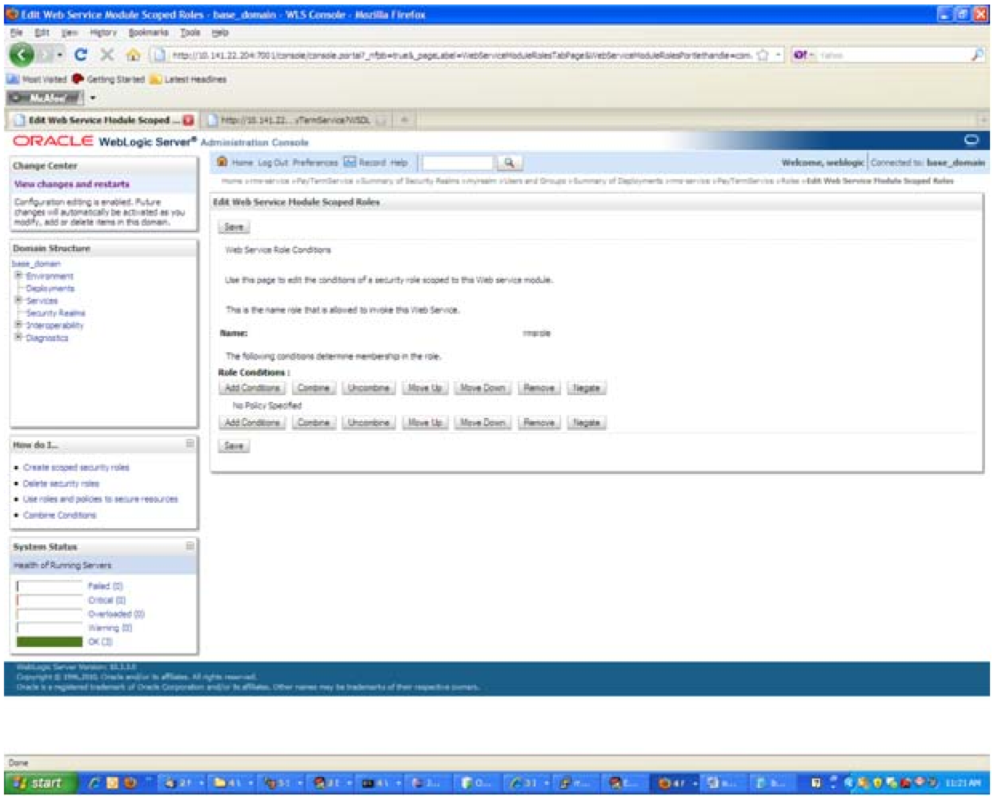
Customize this table

Web Service Module Scoped Roles

New | Delete — Showing 1 to 1 of 1 Previous | Next

Name	Provider Name
mrsrole	XACMLRoleMapper

10. 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.



Edit Web Service Module Scoped Roles - base_domain - WLS Console - Mozilla Firefox

Edit Web Service Module Scoped Roles

Save

Web Service Role Conditions

Use this page to edit the conditions of a security role scoped to this Web service module.

This is the name role that is allowed to invoke this Web Service.

Name: mrsrole

The following conditions determine membership in the role.

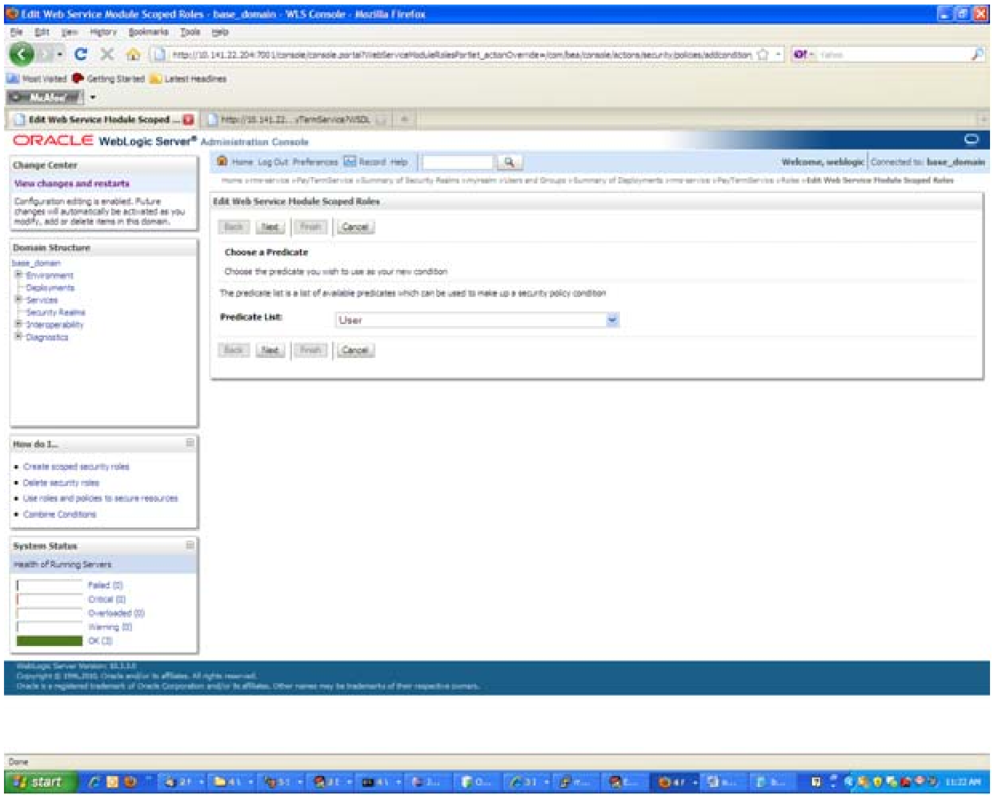
Role Conditions :

Add Conditions | Combine | Uncombine | Move Up | Move Down | Remove | Negate

No Policy Specified

Save

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



Edit Web Service Module Scoped Roles

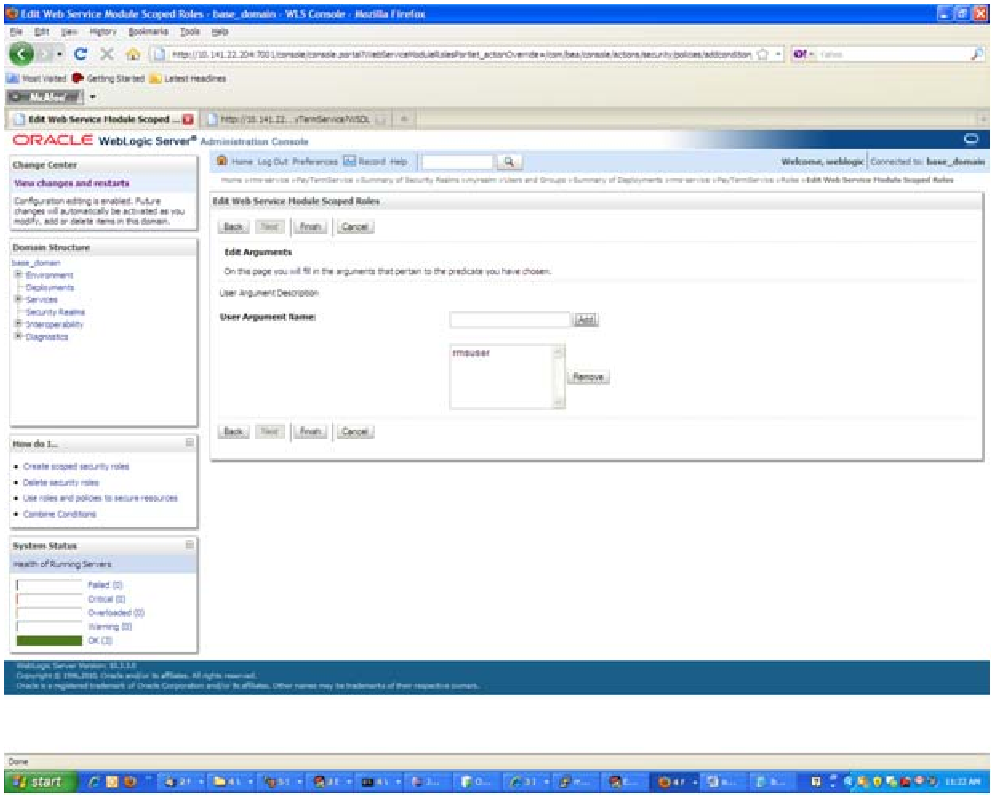
Choose a Predicate

Choose the predicate you wish to use as your new condition

The predicate list is a list of available predicates which can be used to make up a security policy condition

Predicate List: User

12. From Predicate List, select User. Click **Next**. An option to Edit Arguments is displayed.



Edit Web Service Module Scoped Roles

Edit Arguments

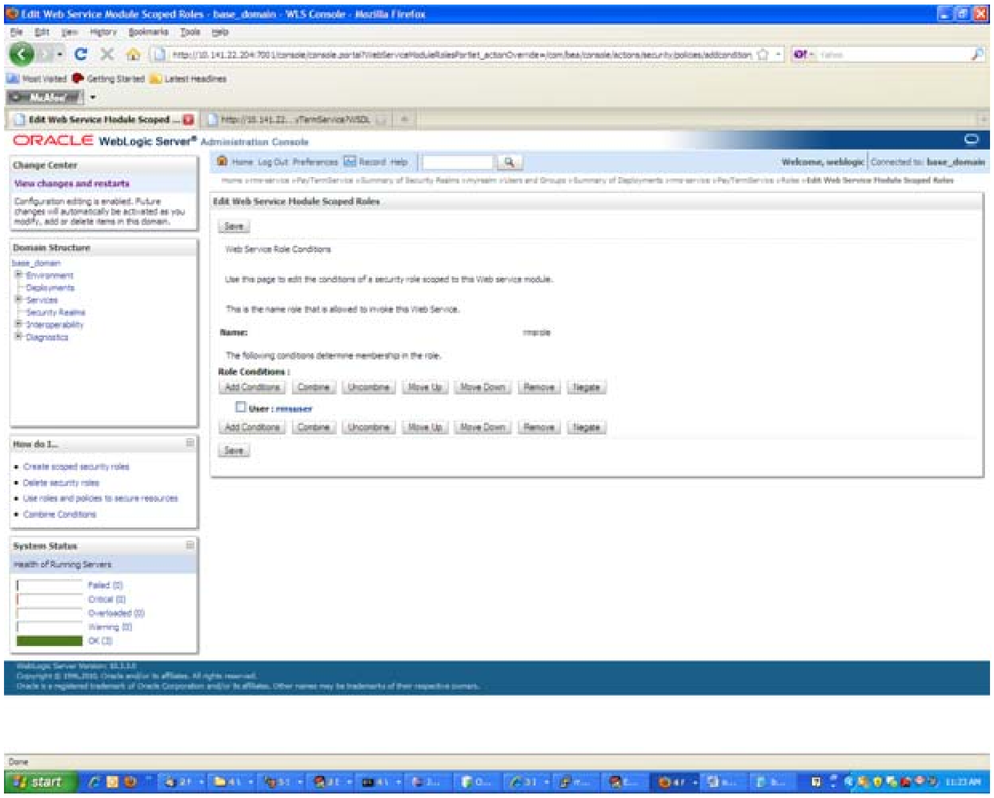
On this page you will fill in the arguments that pertain to the predicate you have chosen.

User Argument Description

User Argument Name:

rmsuser

13. In the User Argument Name field, enter the user name 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.


Edit Web Service Module Scoped Roles - base_domain - WLS Console - Mozilla Firefox
ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Change Center
View changes and restarts
Domain Structure
base_domain
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
How do I...
Create scoped security roles
Delete security roles
Use roles and policies to secure resources
Combine Conditions
System Status
Health of Running Servers
Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
OK (3)
Save
Add Conditions
Combine
Uncombine
Move Up
Move Down
Remove
Negate
start

Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

Edit Web Service Module Scoped Roles

Web Service Role Conditions

Use this page to edit the conditions of a security role scoped to this Web service module.

This is the name role that is allowed to invoke this Web Service.

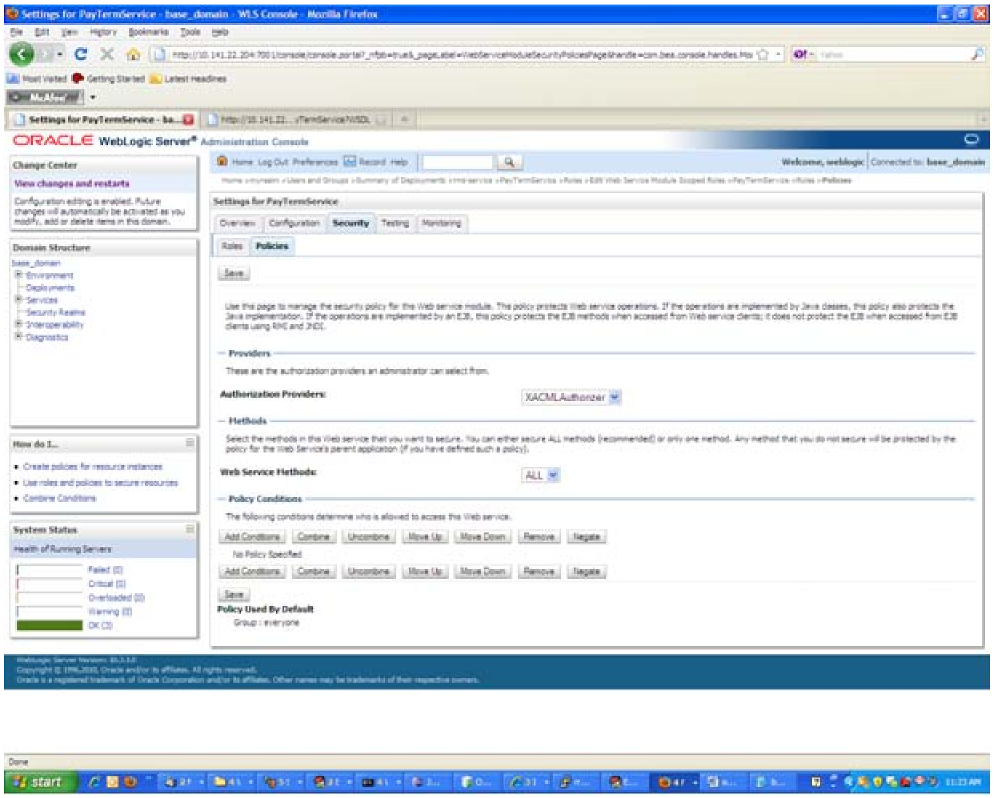
Name: rmsrole

The following conditions determine membership in the role.

Role Conditions :

User : rmsuser

14. Click **Save**. The same screen is displayed with this message near the top: "Changes saved successfully."
15. Return to the Security tab of the Web service and click the **Policies** tab.



Settings for PayTermService

Use this page to manage the security policy for the Web service module. This policy protects Web service operations. If the operations are implemented by Java classes, this policy also protects the Java implementation. If the operations are implemented by an EJB, this policy protects the EJB methods when accessed from Web service clients; it does not protect the EJB when accessed from EJB clients using RMI and JNDI.

Providers

These are the authorization providers an administrator can select from.

Authorization Providers: XACMLAuthorizer

Methods

Select the methods in this Web service 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 Web Service's parent application (if you have defined such a policy).

Web Service Methods: ALL

Policy Conditions

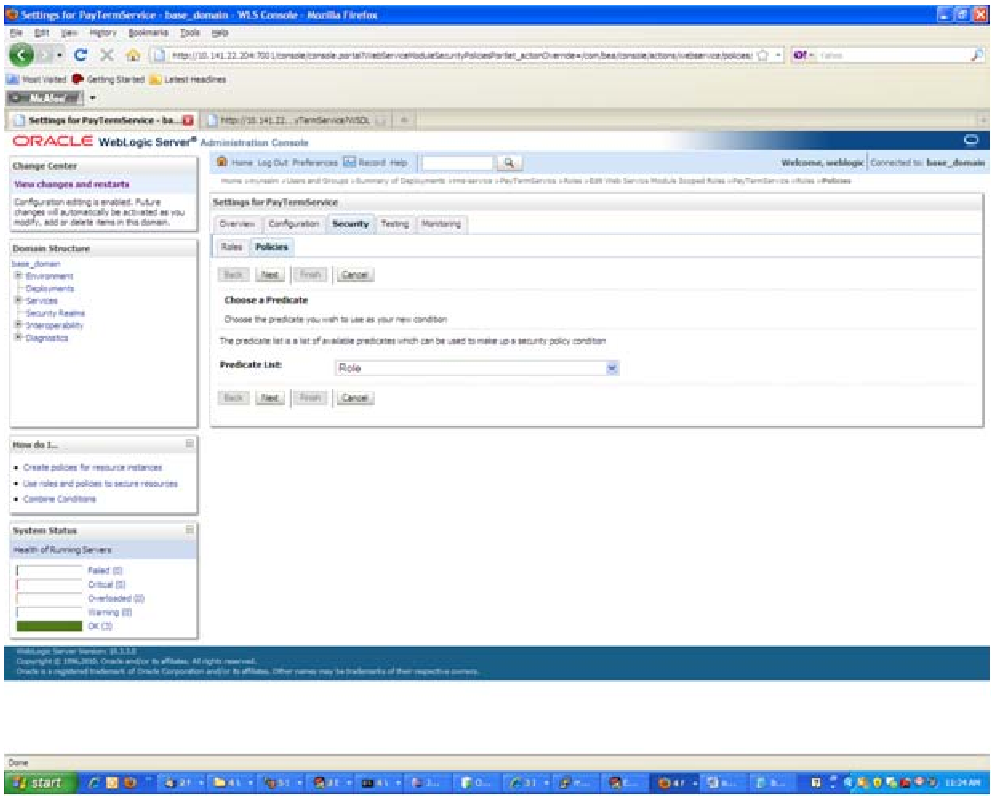
The following conditions determine who is allowed to access the Web service.

No Policy Specified

Policy Used By Default

Group : everyone

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



Settings for PayTermService

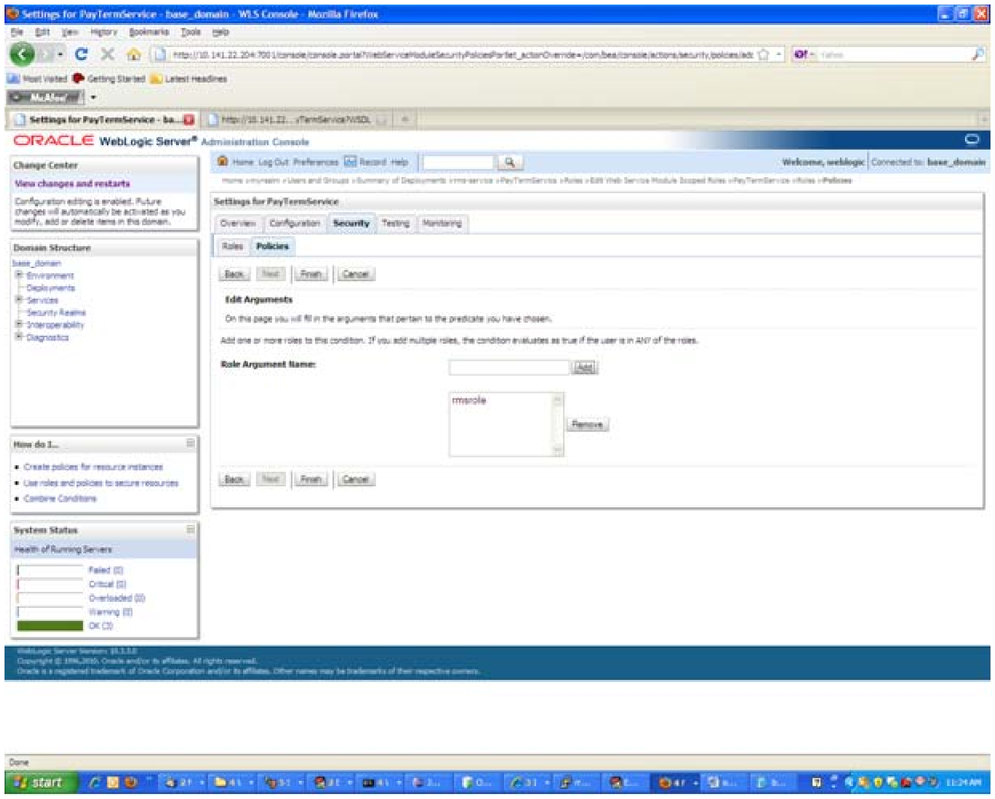
Choose a Predicate

Choose the predicate you wish to use as your new condition

The predicate list is a list of available predicates which can be used to make up a security policy condition

Predicate List: Role

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



Settings for PayTermService

Edit Arguments

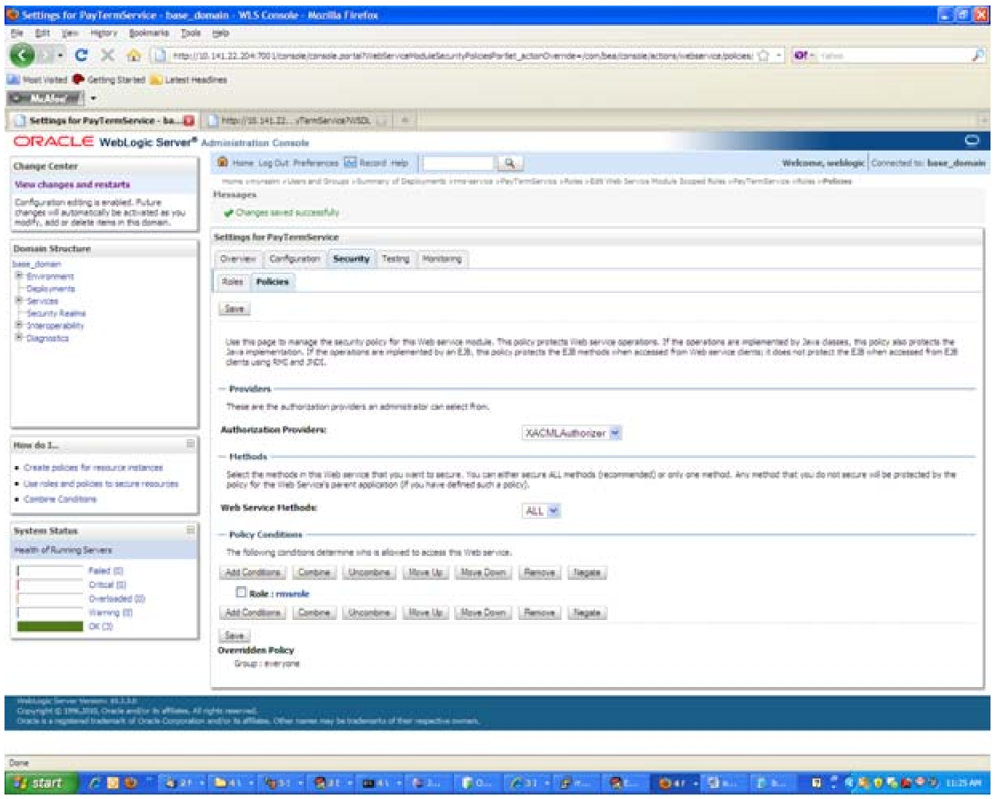
On this page you will fill in the arguments that pertain to the predicate you have chosen.

Add one or more roles to this condition. If you add multiple roles, the condition evaluates as true if the user is in ANY of the roles.

Role Argument Name:

rmsrole

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



Settings for PayTermService

Messages

Changes saved successfully

Overview | Configuration | Security | Testing | Monitoring

Roles | Policies

Use this page to manage the security policy for this Web service module. This policy protects Web service operations. If the operations are implemented by Java classes, this policy also protects the Java implementation. If the operations are implemented by an EJB, the policy protects the EJB methods when accessed from Web service clients; it does not protect the EJB when accessed from EJB clients using RMI and JNDI.

Providers

These are the authorization providers an administrator can select from.

Authorization Providers: XACMLAuthorizer

Methods

Select the methods in this Web service 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 Web Service's parent application (if you have defined such a policy).

Web Service Methods: ALL

Policy Conditions

The following conditions determine who is allowed to access this Web service.

Role : rmsrole

Overridden Policy

Group : everyone

Client-Side Setup

Web services can be invoked from Java clients as well as PL/SQL clients. This section describes the configuration for invoking a secured Web service from both clients.

Java Client Setup

Client code for calling Web services can be generated using the Java consumer option of the retail-soa-enabler-gui tool. The generated zip file contains all the jar files required for the classpath of the application that calls the Web service. To run the client, follow the steps required to run Java consumer.

The following is sample code for calling a secured Web service.

Note: The code below is sample code for invoking the PayTerm service. When you generate Java consumer for a Web service, the generated jar file will contain classes specific to that Web service. Use the appropriate classes in the client code. Service namespace and WSDL location also should be changed appropriately.

```
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.integration.base.bo.paytermdesc.v1.PayTermDesc;
import com.oracle.retail.integration.base.bo.paytermref.v1.PayTermRef;
import
com.oracle.retail.rms.integration.services.paytermservice.v1.PayTermPortType;
import
com.oracle.retail.rms.integration.services.paytermservice.v1.PayTermService;

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 PayTermClient extends TestCase{
       public void testFindPayTerm(){
              try{
                      //qName is namespace of the service
                      QName qName = new
QName("http://www.oracle.com/retail/rms/integration/services/PayTermService/v1","P
ayTermService");

                      // url is the URL of the WSDL of the web service
                      URL url = new
URL("http://10.141.26.93:7001/PayTermBean/PayTermService?WSDL");

                     //create an instance of the web service
                     PayTermService service = new PayTermService(url,qName);
                     PayTermPortType port =   service.getPayTermPort();

                     //set the security credentials in the service context
                     List credProviders = new ArrayList();
                     CredentialProvider cp = new
ClientUNTCredentialProvider("rmsuser","rmsuser1");
                     credProviders.add(cp);
                     Map<String, Object> rc =
((BindingProvider)port).getRequestContext();
                     rc.put(WSSecurityContext.CREDENTIAL_PROVIDER_LIST,
credProviders);

                     //populate the service method input object
                    PayTermRef ref = new PayTermRef();
                    ref.setTerms("terms");
                    ref.setTermsXrefKey("key");

                    //call the web service.here desc is the response object
                    PayTermDesc desc =  port.findPayTermDesc(ref);

                    System.out.println("desc="+desc);
            }catch(Exception e){
                   e.printStackTrace();
            }
      }
}
```

PL/SQL Client Setup

Client code for calling Web services can be generated using the pl/sql consumer option of the retail-soa-enabler-gui tool. The generated zip file contains all the jar files and pl/sql code required to invoke the web service from pl/sql. To run the client, follow the steps required to run pl/sql consumer.

The following is a sample PL/SQL procedure for calling a secured Web service.

Note: The code below is sample code for invoking the PayTerm service. When you generate PL/SQL consumer for a Web service, the generated jar file will contain classes specific to that Web service. Use the appropriate classes in the client code. Service namespace and WSDL location also should be changed appropriately.

```
create or replace
PROCEDURE wstestClient IS

ref "OBJ_PAYTERMREF" := null ;
objdesc "OBJ_PAYTERMDESC" := null;
wsm varchar2(100);config varchar2(32000);
BEGIN
wsm := 'oracle.webservices.dii.interceptor.pipeline.port.config';
   config :='<port-info>
   <runtime enabled="security">
   <security>
   <outbound>
   <username-token name="" password=""/>
   </outbound>
   </security>
   </runtime>
   </port-info>' ;

 PayTermServiceConsumer.setProperty(wsm, config);

PayTermServiceConsumer.setEndpoint('http://10.141.22.204:7001/PayTermBean/PayTermS
ervice');
PayTermServiceConsumer.setUsername('rmsuser');
PayTermServiceConsumer.setPassword('rmsuser2');
ref := "OBJ_PAYTERMREF"('x','t',null,null,null);
dbms_output.PUT_LINE(PayTermServiceConsumer.getEndPoint());
dbms_output.PUT_LINE(PayTermServiceConsumer.ping('TestMessage'));
objdesc := PayTermServiceConsumer.findPayTermDesc(ref) ;
dbms_output.PUT_LINE('Done.');
EXCEPTION
   WHEN OTHERS THEN
    dbms_output.PUT_LINE(SQLCODE);
   dbms_output.PUT_LINE(SQLERRM);
END;
```

Server-Side Setup for Encrypted User Name and Password Token Authentication

There are many predefined policy files provided by WebLogic that can be used for securing Web services. This section describes the process required to secure a Web service where user name and password are encrypted and signed.

The steps to attach the policy file to the Web service are smiliar to the steps described in the section, "Attach Policy File to the Web Service." The only difference is that for the Step 7, policy:Wssp1.2-2007-Wss1.1-UsernameToken-Plain-X509-Basic256.xml, should be selected. The rest of the steps should be followed exactly as they are.

After attaching the policy file, the WSDL of the Web service will include the following content in the header:

```
<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/Includ
eToken/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/Includ
eToken/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/Includ
eToken/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>
```

For this policy to work the server has to be able to trust that the key combination used by the client to sign the message is valid. To do this, the client certificate must be signed with a certificate authority that is trusted by the server.

WebLogic instances come with a demo CA. The certificate and key for this are in $WL_HOME/Middleware/wlserver_10.3/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 it is very important to never have the DemoTrust store enabled in a production environment. Otherwise, anyone can become trusted fairly easily. Setting up a development environment it is useful.

WebLogic CertGen command can be used to generate keys of the correct length and sign it with the demo CA. A client cert/key pair is required to sign the outgoing message and the server certificate to encrypt the critical portions.

java -classpath $WL_HOME/Middleware/wlserver_10.3/server/lib/weblogic.jar utils.CertGen -certfile ClientCert -keyfile ClientKey -keyfilepass ClientKey -cn rmsuser

This generates the following files. The user name, "rmsuser," should be replaced with the user name, which is same as the user who accesses the Web service:

- ClientCert.der
- ClientCert.pem
- ClientKey.der
- ClientKey.pem

```
java -classpath $WL_HOME/Middleware/wlserver_10.3/server/lib/weblogic.jar
utils.CertGen -certfile ServerCert -keyfile ServerKey -keyfilepass ServerKey  -cn
rmsuser
```

This generates the following files. The user name, "rmsuser," should be replaced with the user name, which is same as the user who accesses the Web service:

- ServerCert.der
- ServerCert.pem
- ServerKey.der
- ServerKey.pem

Import these files into key stores using the following commands.

```
java -classpath $WL_HOME/Middleware/wlserver_10.3/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_10.3/server/lib/weblogic.jar
utils.ImportPrivateKey -certfile ServerCert.der -keyfile ServerKey.der
-keyfilepass ServerKey -keystore ServerIdentity.jks -storepass ServerKey -alias
identity -keypass ServerKey
```

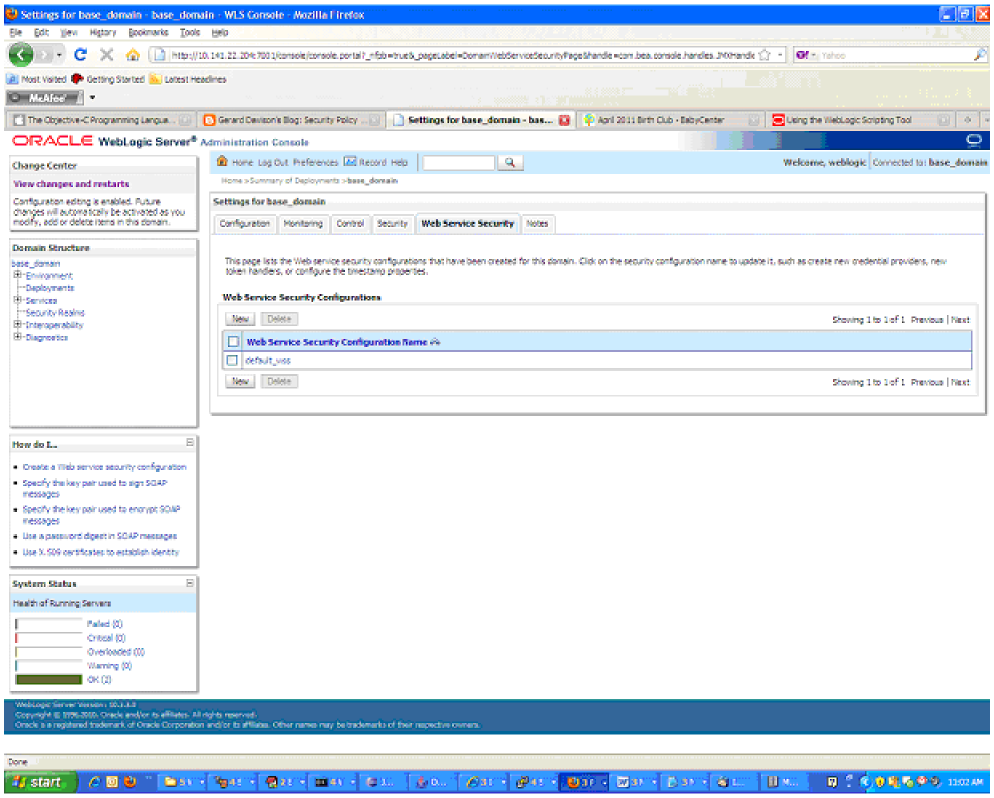
Configure the WebLogic server to use these keys. To configure the server, run the script from the section, "Reference: configWss.py." Copy the script and save it in the location from which it will run.

```
Java -classpath $WL_HOME/Middleware/wlserver_10.3/server/lib/weblogic.jar
weblogic.WLST configWss.py <weblogicuser> <weblogicpassword> <weblogichost>
<weblogic admin port> ServerIdentity.jks ServerKey identity ServerKey
```

For example:

```
Java -classpath $WL_HOME/Middleware/wlserver_10.3/server/lib/weblogic.jar
weblogic.WLST configWss.py weblogic weblogic1 localhost 7001
/home/wls/ServerIdentity.jks ServerKey identity ServerKey
```

Verify that this command has run properly by looking at the "Web Service Security" tab on your domain from the WebLogic console. Note that the default_ww configuration is used for all Web services, unless otherwise indicated.


Settings for base_domain - base_domain - WLS Console - Mozilla Firefox
ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Home >Summary of Deployments >base_domain
Change Center
View changes and restarts
Domain Structure
base_domain
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
How do I...
System Status
Health of Running Servers
Configuration Monitoring Control Security Web Service Security Notes
New Delete

Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

Settings for base_domain

This page lists the Web service security configurations that have been created for this domain. Click on the security configuration name to update it, such as create new credential providers, new token handlers, or configure the timestamp properties.

Web Service Security Configurations

Showing 1 to 1 of 1 Previous | Next

Web Service Security Configuration Name
default_wss

- 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

Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
OK (2)

WebLogic Server Version: 10.3.3.0
Copyright © 1996-2010, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

After the certificate setup for the Web service is finished, create a user in WebLogic with access to the Web service. See Create Roles and Users for information.

After restarting the server, a client can be created to invoke the Web service.

Client-Side Setup for Encrypted User Name and Password Token Authentication

The following is sample code for calling a Web service that has been secured using this 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.integration.base.bo.locofpaytermref.v1.LocOfPayTermRef;
import com.oracle.retail.integration.base.bo.paytermdesc.v1.PayTermDesc;
import com.oracle.retail.integration.base.bo.paytermref.v1.PayTermRef;
import com.oracle.retail.integration.custom.bo.extofpaytermref.v1.ExtOfPayTermRef;
import
com.oracle.retail.rms.integration.services.paytermservice.v1.PayTermPortType;
import
com.oracle.retail.rms.integration.services.paytermservice.v1.PayTermService;

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{
PayTermService service = new PayTermService(new
URL("http://10.141.22.204:7001/PayTermBean/PayTermService?WSDL"), new
QName("http://www.oracle.com/retail/rms/integration/services/PayTermService/v1",
"PayTermService"));
PayTermPortType port =  service.getPayTermPort();
PayTermRef ref = new PayTermRef();
ref.setTerms("t");
ref.setTermsXrefKey("x");
ExtOfPayTermRef e = new ExtOfPayTermRef();

ref.setExtOfPayTermRef(e);
LocOfPayTermRef l = new LocOfPayTermRef();

ref.setLocOfPayTermRef(l);
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("rmsuser",
"rmsuser1");
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;
}
}
);
PayTermDesc desc =  port.findPayTermDesc(ref);
System.out.println("desc="+desc);
}catch(Exception e){
e.printStackTrace();
}
}
}
```

Reference: configWss.py

```
userName = sys.argv[1]
passWord = sys.argv[2]
url="t3://"+sys.argv[3]+":"+sys.argv[4]
print "Connect to the running adminSever"
connect(userName, passWord, url)
edit()
startEdit()
#Enable assert x509 in SecurityConfiguration
rlm = cmo.getSecurityConfiguration().getDefaultRealm()
ia = rlm.lookupAuthenticationProvider("DefaultIdentityAsserter")
activeTypesValue = list(ia.getActiveTypes())
existed = "X.509" in activeTypesValue
if existed == 1:
  print 'assert x509 is aleady enabled'
else:
  activeTypesValue.append("X.509")
ia.setActiveTypes(array(activeTypesValue,java.lang.String))
ia.setDefaultUserNameMapperAttributeType('CN');
ia.setUseDefaultUserNameMapper(Boolean('true'));

#Create default WebServcieSecurity

securityName='default_wss'
defaultWss=cmo.lookupWebserviceSecurity(securityName)
if defaultWss == None:
  print 'creating new webservice security bean for: ' + securityName
  defaultWss = cmo.createWebserviceSecurity(securityName)
else:
  print 'found exsiting bean for: ' + securityName

#Create credential provider for DK
```

```
cpName='default_dk_cp'
wtm=defaultWss.lookupWebserviceCredentialProvider(cpName)
if wtm == None:
wtm = defaultWss.createWebserviceCredentialProvider(cpName)
wtm.setClassName('weblogic.wsee.security.wssc.v200502.dk.DKCredentialProvider')
wtm.setTokenType('dk')
cpm = wtm.createConfigurationProperty('Label')
cpm.setValue('WS-SecureConversationWS-SecureConversation')
cpm = wtm.createConfigurationProperty('Length')
cpm.setValue('16')

else:
  print 'found exsiting bean for: DK ' + cpName

#Create credential provider for x.509

cpName='default_x509_cp'
wtm=defaultWss.lookupWebserviceCredentialProvider(cpName)
if wtm == None:
wtm = defaultWss.createWebserviceCredentialProvider(cpName)
wtm.setClassName('weblogic.wsee.security.bst.ServerBSTCredentialProvider')
wtm.setTokenType('x509')
else:
  print 'found exsiting bean for: x.509 ' + cpName


#Custom keystore for xml encryption

cpName='ConfidentialityKeyStore'
cpm=wtm.lookupConfigurationProperty(cpName)
if cpm == None:
cpm = wtm.createConfigurationProperty(cpName)
keyStoreName=sys.argv[5]
cpm.setValue(keyStoreName)
cpName='ConfidentialityKeyStorePassword'
cpm=wtm.lookupConfigurationProperty(cpName)
if cpm == None:
cpm = wtm.createConfigurationProperty(cpName)
cpm.setEncryptValueRequired(Boolean('true'))
KeyStorePasswd=sys.argv[6]
cpm.setEncryptedValue(KeyStorePasswd)
cpName='ConfidentialityKeyAlias'
cpm=wtm.lookupConfigurationProperty(cpName)
if cpm == None:
cpm = wtm.createConfigurationProperty(cpName)
keyAlias=sys.argv[7]
cpm.setValue(keyAlias)
cpName='ConfidentialityKeyPassword'
cpm=wtm.lookupConfigurationProperty(cpName)
if cpm == None:
cpm = wtm.createConfigurationProperty('ConfidentialityKeyPassword')
cpm.setEncryptValueRequired(Boolean('true'))
keyPass=sys.argv[8]
cpm.setEncryptedValue(keyPass)

#Custom keystore for xml digital signature

cpName='IntegrityKeyStore'
cpm=wtm.lookupConfigurationProperty(cpName)
```

```
if cpm == None:
cpm = wtm.createConfigurationProperty(cpName)
keyStoreName=sys.argv[5]
cpm.setValue(keyStoreName)
cpName='IntegrityKeyStorePassword'
cpm=wtm.lookupConfigurationProperty(cpName)
if cpm == None:
cpm = wtm.createConfigurationProperty(cpName)
cpm.setEncryptValueRequired(Boolean('true'))
KeyStorePasswd=sys.argv[6]
cpm.setEncryptedValue(KeyStorePasswd)
cpName='IntegrityKeyAlias'
cpm=wtm.lookupConfigurationProperty(cpName)
if cpm == None:
cpm = wtm.createConfigurationProperty(cpName)
keyAlias=sys.argv[7]
cpm.setValue(keyAlias)
cpName='IntegrityKeyPassword'
cpm=wtm.lookupConfigurationProperty(cpName)
if cpm == None:
cpm = wtm.createConfigurationProperty(cpName)
cpm.setEncryptValueRequired(Boolean('true'))
keyPass=sys.argv[8]
cpm.setEncryptedValue(keyPass)

#Create token handler for x509 token

#cpName='default_x509_handler'
th=defaultWss.lookupWebserviceTokenHandler(cpName)
if th == None:
th = defaultWss.createWebserviceTokenHandler(cpName)
th.setClassName('weblogic.xml.crypto.wss.BinarySecurityTokenHandler')
th.setTokenType('x509')
cpm = th.createConfigurationProperty('UseX509ForIdentity')
cpm.setValue('true')
save()
activate(block="true")
disconnect()
exit()
```

A

Appendix: Installer Screens

This appendix provides step-by-step instructions for installing the Oracle Retail Service-Oriented Architecture Enabler tool as a Web application in Oracle WebLogic.

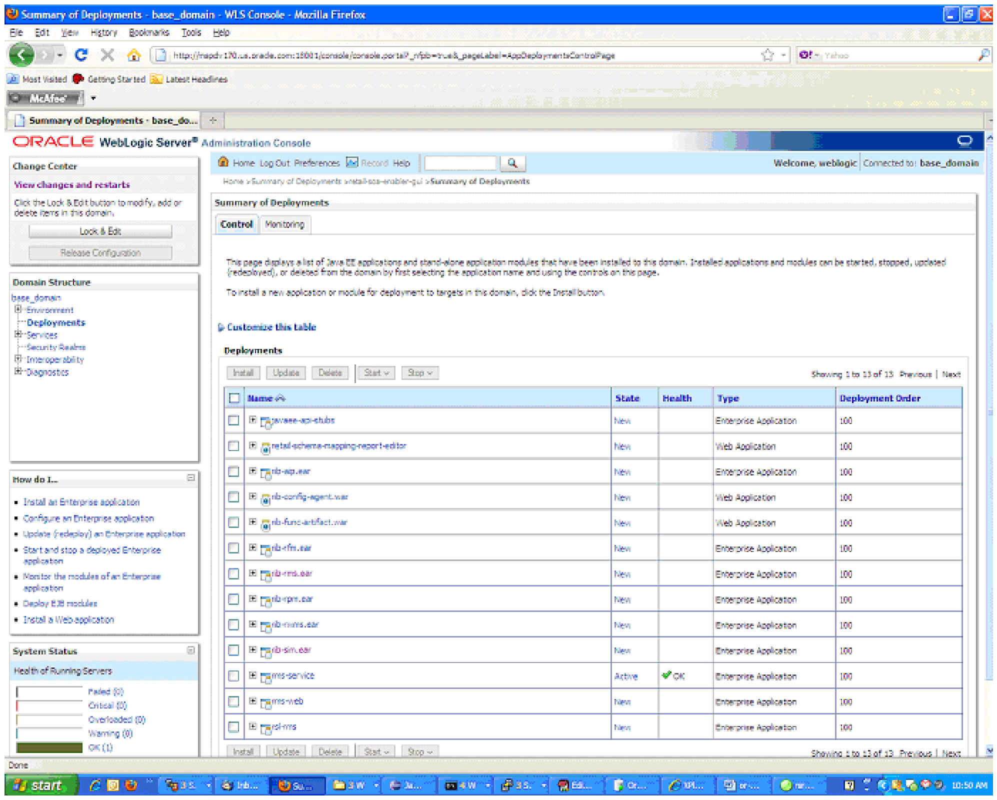
Installation as a Web Application in Oracle WebLogic

To install the Oracle Retail Service-Oriented Architecture Enabler tool as a Web application in Oracle WebLogic, complete these steps.

Deploy the Retail SOA Enabler Application

Using the WebLogic Server Administration Console, complete the following steps:

1. Navigate to the Deployments page:



Summary of Deployments

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

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

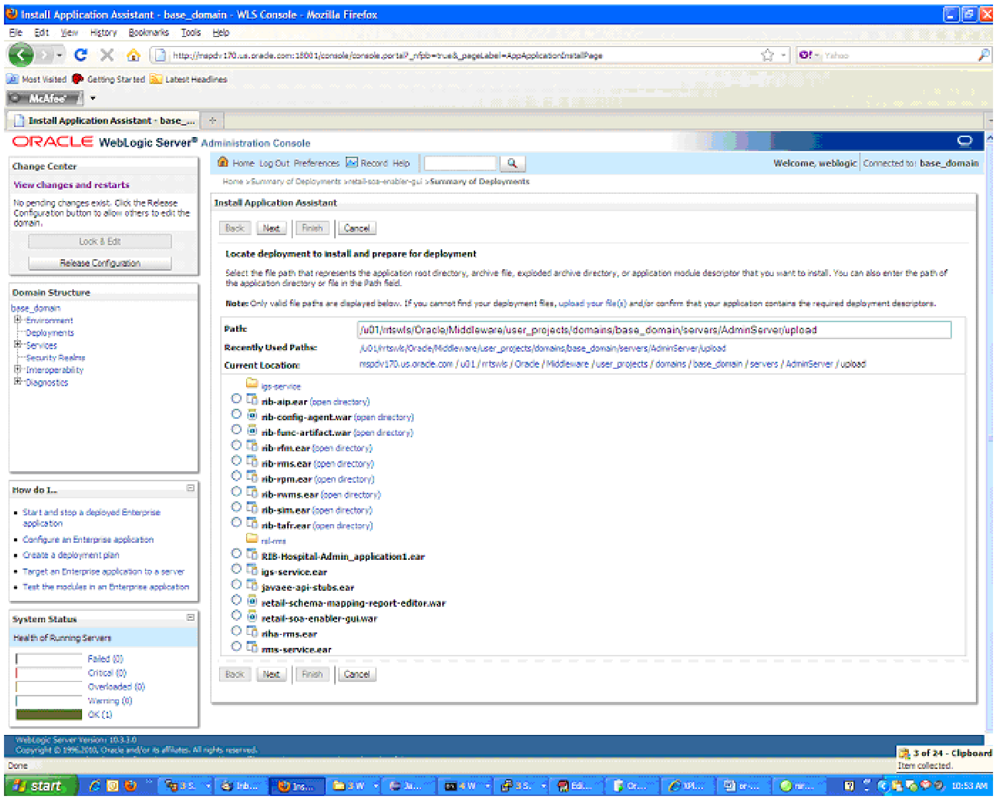
Deployments

Showing 1 to 13 of 13 Previous | Next

Name	State	Health	Type	Deployment Order
javaee-api-stubs	New		Enterprise Application	100
retail-schema-mapping-report-editor	New		Web Application	100
rib-aip.ear	New		Enterprise Application	100
rib-config-agent.war	New		Web Application	100
rib-func-artifact.war	New		Web Application	100
rib-rfm.ear	New		Enterprise Application	100
rib-rms.ear	New		Enterprise Application	100
rib-rpm.ear	New		Enterprise Application	100
rib-rwms.ear	New		Enterprise Application	100
rib-sim.ear	New		Enterprise Application	100
rms-service	Active	✓ OK	Enterprise Application	100
rms-web	New		Enterprise Application	100
rsl-rms	New		Enterprise Application	100

Showing 1 to 13 of 13 Previous | Next

2. In the left navigation bar, click **Lock & Edit**. Click **Install**.



Install Application Assistant - base_domain - WLS Console - Mozilla Firefox

ORACLE WebLogic Server® Administration Console

Welcome, weblogic | Connected to: base_domain

Home >Summary of Deployments >retail-soa-enabler-gui >Summary of Deployments

Change Center

View changes and restarts

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

Domain Structure

base_domain
- Environment
- Deployments
- Services
- Security Realms
- Interoperability
- Diagnostics

How do I...

- Start and stop a deployed Enterprise application
- Configure an Enterprise application
- Create a deployment plan
- Target an Enterprise application to a server
- Test the modules in an Enterprise application

System Status

Health of Running Servers

- Failed (0)
- Critical (0)
- Overloaded (0)
- Warning (0)
- OK (1)

Install Application Assistant

Locate deployment to install and prepare for deployment

Select the file path that represents the application root directory, archive file, exploded archive directory, or application module descriptor that you want to install. You can also enter the path of the application directory or file in the Path field.

Note: Only valid file paths are displayed below. If you cannot find your deployment files, upload your file(s) and/or confirm that your application contains the required deployment descriptors.

Path:	/u01/rrtswls/Oracle/Middleware/user_projects/domains/base_domain/servers/AdminServer/upload
Recently Used Paths:	/u01/rrtswls/Oracle/Middleware/user_projects/domains/base_domain/servers/AdminServer/upload
Current Location:	mspdv170.us.oracle.com / u01 / rrtswls / Oracle / Middleware / user_projects / domains / base_domain / servers / AdminServer / upload

- igs-service
- rib-aip.ear (open directory)
- rib-config-agent.war (open directory)
- rib-func-artifact.war (open directory)
- rib-rfm.ear (open directory)
- rib-rms.ear (open directory)
- rib-rpm.ear (open directory)
- rib-rwms.ear (open directory)
- rib-sim.ear (open directory)
- rib-tafr.ear (open directory)
- rsl-rms
- RIB-Hospital-Admin_application1.ear
- igs-service.ear
- javaee-api-stubs.ear
- retail-schema-mapping-report-editor.war
- retail-soa-enabler-gui.war
- riha-rms.ear
- rms-service.ear

WebLogic Server Version: 10.3.3.0
Copyright © 1996,2010, Oracle and/or its affiliates. All rights reserved.

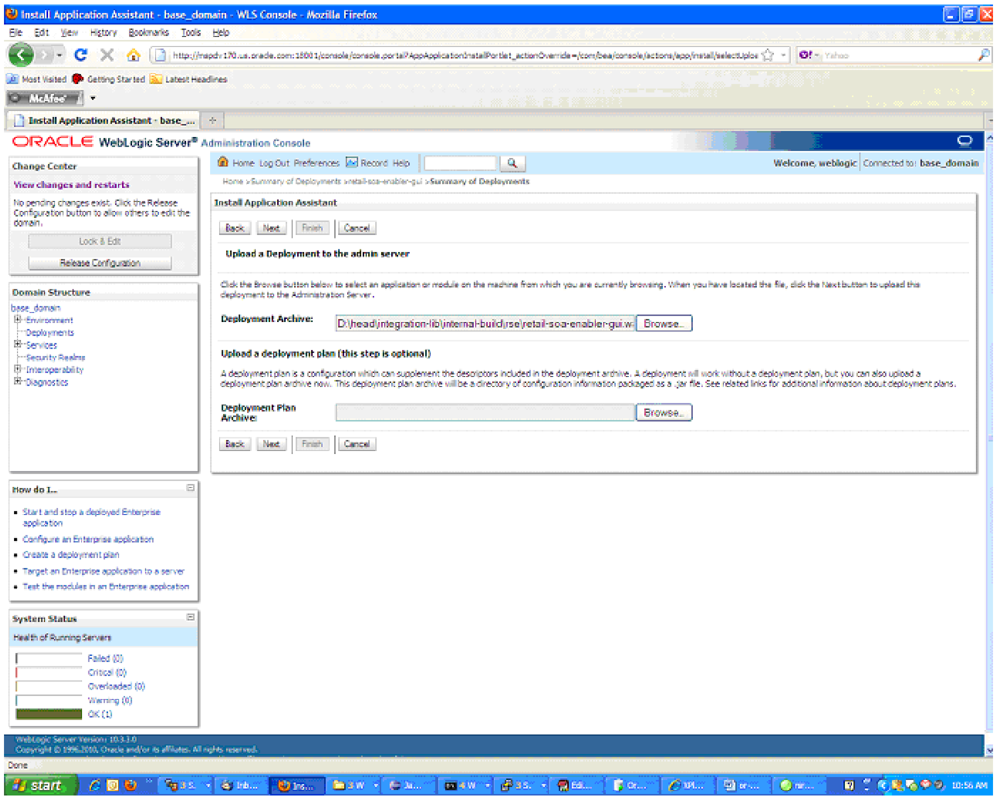
File Edit View History Bookmarks Tools Help
http://mspdv170.us.oracle.com:18001/console/console.portal?_nfpb=true&_pageLabel=AppApplicationInstallPage
Most Visited
Getting Started
Latest Headlines
McAfee
Install Application Assistant - base_...
Home Log Out Preferences Record Help
Lock & Edit
Release Configuration
Back
Next
Finish
Cancel
3 of 24 - Clipboard
Item collected.
Done
start
10:53 AM

Note: If the application has already been installed, see the section, "Redeploy the Application."

The **Locate deployment to install and prepare for deployment page** is displayed. Follow the instructions to locate the retail-soa-enabler-gui.war file.

3. Select **Upload Files**.

4. On the **Upload a Deployment to the admin server** page, use the **Browse** button to locate the retail-soa-enabler-gui.war file in the Deployment Archive.


Install Application Assistant - base_domain - WLS Console - Mozilla Firefox

Home >Summary of Deployments >install-soa-enabler-gui >Summary of Deployments

Welcome, weblogic | Connected to: base_domain

Change Center

View changes and restarts

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

Domain Structure

base_domain
- Environment
- Deployments
- Services
- Security Realms
- Interoperability
- Diagnostics

Install Application Assistant

Upload a Deployment to the admin server

Click the Browse button below to select an application or module on the machine from which you are currently browsing. When you have located the file, click the Next button to upload this deployment to the Administration Server.

Deployment Archive: D:\head\integration-lib\internal-build\rse\retail-soa-enabler-gui.w

Upload a deployment plan (this step is optional)

A deployment plan is a configuration which can supplement the descriptors included in the deployment archive. A deployment will work without a deployment plan, but you can also upload a deployment plan archive now. This deployment plan archive will be a directory of configuration information packaged as a .jar file. See related links for additional information about deployment plans.

Deployment Plan Archive:

How do I...
- Start and stop a deployed Enterprise application
- Configure an Enterprise application
- Create a deployment plan
- Target an Enterprise application to a server
- Test the modules in an Enterprise application

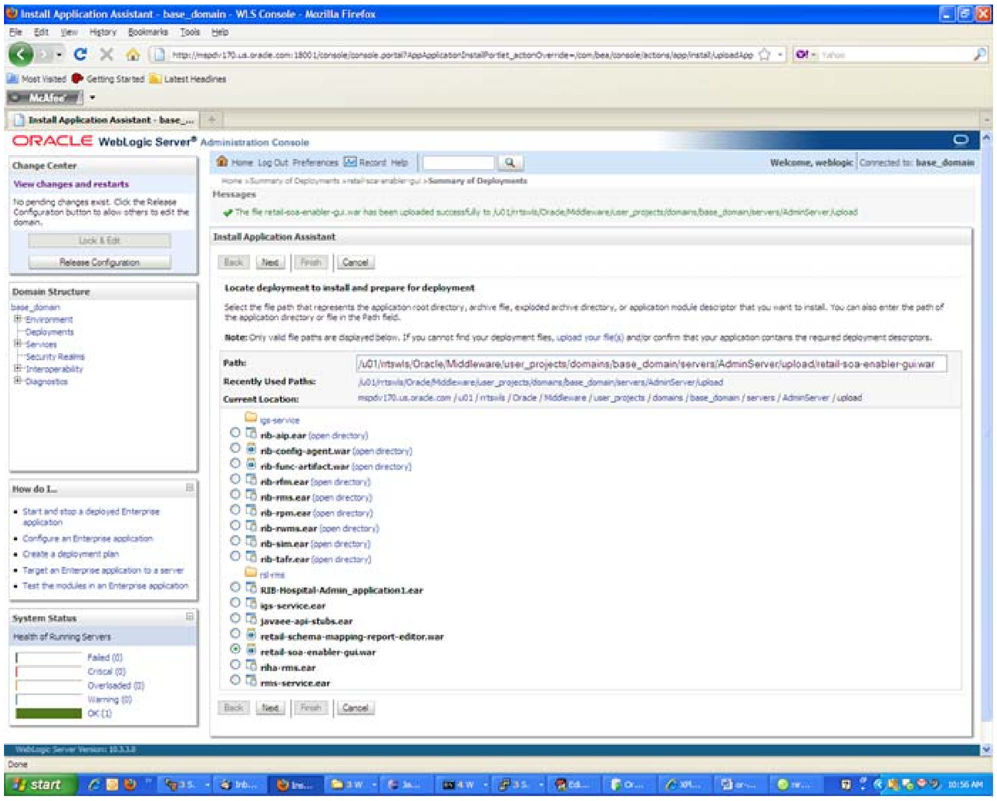
System Status

Health of Running Servers

Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
OK (1)

WebLogic Server Version: 10.3.3.0
Copyright © 1996,2010, Oracle and/or its affiliates. All rights reserved.

5. Select the retail-soa-enabler-gui.war.


Install Application Assistant - base_domain - WLS Console - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Most Visited Getting Started Latest Headlines
ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Back Next Finish Cancel

Home >Summary of Deployments >retail-soa-enabler-gui >Summary of Deployments

Change Center

View changes and restarts

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

Lock & Edit

Release Configuration

Domain Structure

base_domain
- Environment
- Deployments
- Services
- Security Realms
- Interoperability
- Diagnostics

How do I...

- Start and stop a deployed Enterprise application
- Configure an Enterprise application
- Create a deployment plan
- Target an Enterprise application to a server
- Test the modules in an Enterprise application

System Status

Health of Running Servers

Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
OK (1)

Messages

✓ The file retail-soa-enabler-gui.war has been uploaded successfully to /u01/rtswls/Oracle/Middleware/user_projects/domains/base_domain/servers/AdminServer/upload

Install Application Assistant

Locate deployment to install and prepare for deployment

Select the file path that represents the application root directory, archive file, exploded archive directory, or application module descriptor that you want to install. You can also enter the path of the application directory or file in the Path field.

Note: Only valid file paths are displayed below. If you cannot find your deployment files, upload your file(s) and/or confirm that your application contains the required deployment descriptors.

Path:	/u01/rtswls/Oracle/Middleware/user_projects/domains/base_domain/servers/AdminServer/upload/retail-soa-enabler-gui.war
Recently Used Paths:	/u01/rtswls/Oracle/Middleware/user_projects/domains/base_domain/servers/AdminServer/upload
Current Location:	mspdv170.us.oracle.com / u01 / rtswls / Oracle / Middleware / user_projects / domains / base_domain / servers / AdminServer / upload

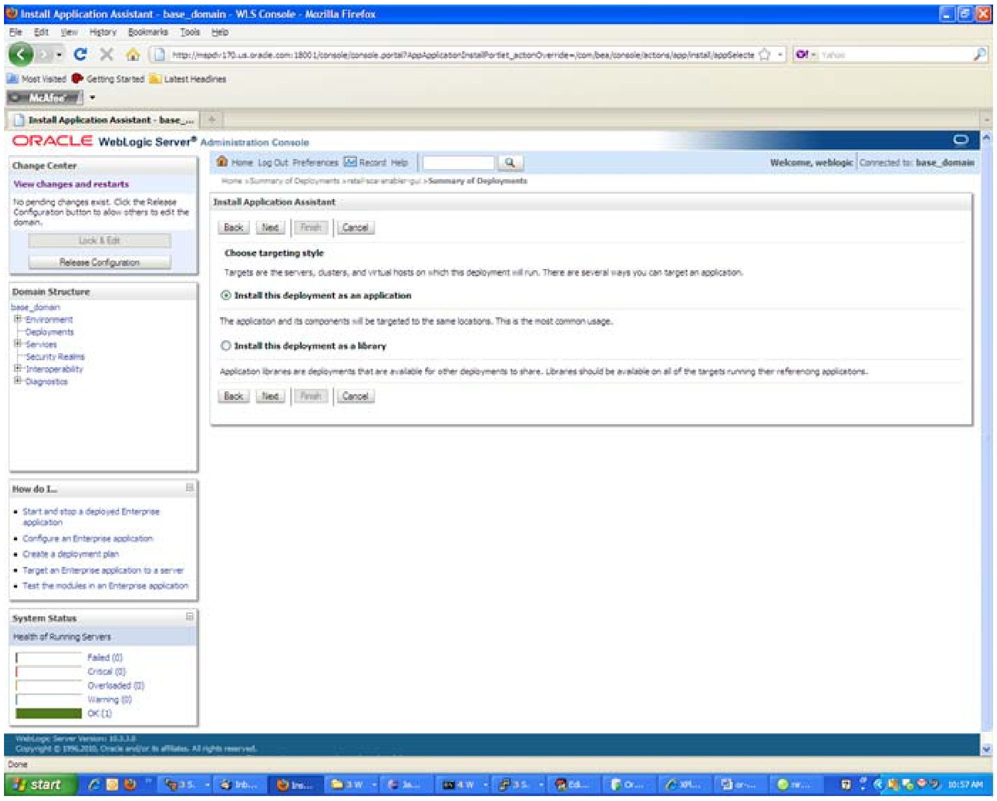
igs-service
- ○ rib-aip.ear (open directory)
- ○ rib-config-agent.war (open directory)
- ○ rib-func-artifact.war (open directory)
- ○ rib-rfm.ear (open directory)
- ○ rib-rms.ear (open directory)
- ○ rib-rpm.ear (open directory)
- ○ rib-rwms.ear (open directory)
- ○ rib-sim.ear (open directory)
- ○ rib-tafr.ear (open directory)

rsl-rms
- ○ RIB-Hospital-Admin_application1.ear
- ○ igs-service.ear
- ○ javaee-api-stubs.ear
- ○ retail-schema-mapping-report-editor.war
- [✓ retail-soa-enabler-gui.war]
- ○ riha-rms.ear
- ○ rms-service.ear

WebLogic Server Version: 10.3.3.0

Done

6. Click **Next** and move to **Choose targeting style**.



Install Application Assistant

Choose targeting style

Targets are the servers, clusters, and virtual hosts on which this deployment will run. There are several ways you can target an application.

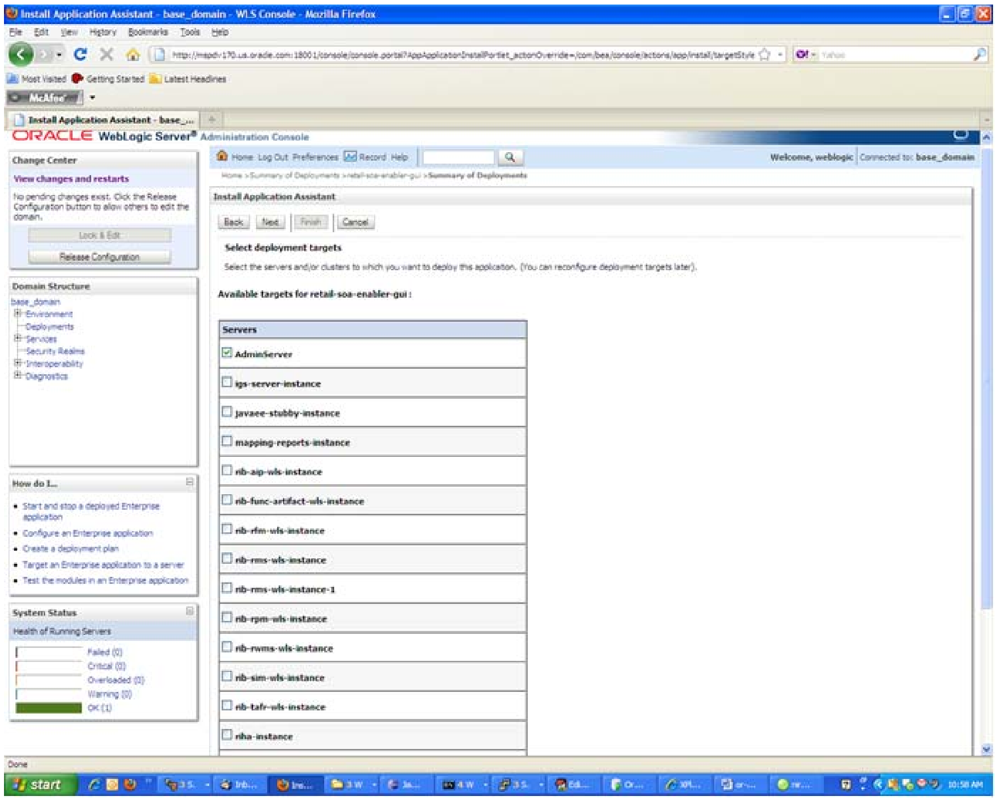
[✓ Install this deployment as an application]

The application and its components will be targeted to the same locations. This is the most common usage.

Install this deployment as a library

Application libraries are deployments that are available for other deployments to share. Libraries should be available on all of the targets running their referencing applications.

7. Select **Install this deployment as an application**.



Install Application Assistant - base_domain - WLS Console - Mozilla Firefox

http://mspdv170.us.oracle.com:18001/console/console.portal?AppApplicationInstallPortlet_actionOverride=/com/bea/console/actions/app/install/targetStyle

Welcome, weblogic | Connected to: base_domain

Home >Summary of Deployments >retail-soa-enabler-gui >Summary of Deployments

Change Center

View changes and restarts

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

Domain Structure

base_domain
- Environment
- Deployments
- Services
- Security Realms
- Interoperability
- Diagnostics

How do I...

- Start and stop a deployed Enterprise application
- Configure an Enterprise application
- Create a deployment plan
- Target an Enterprise application to a server
- Test the modules in an Enterprise application

System Status

Health of Running Servers

- Failed (0)
- Critical (0)
- Overloaded (0)
- Warning (0)
- OK (1)

Install Application Assistant

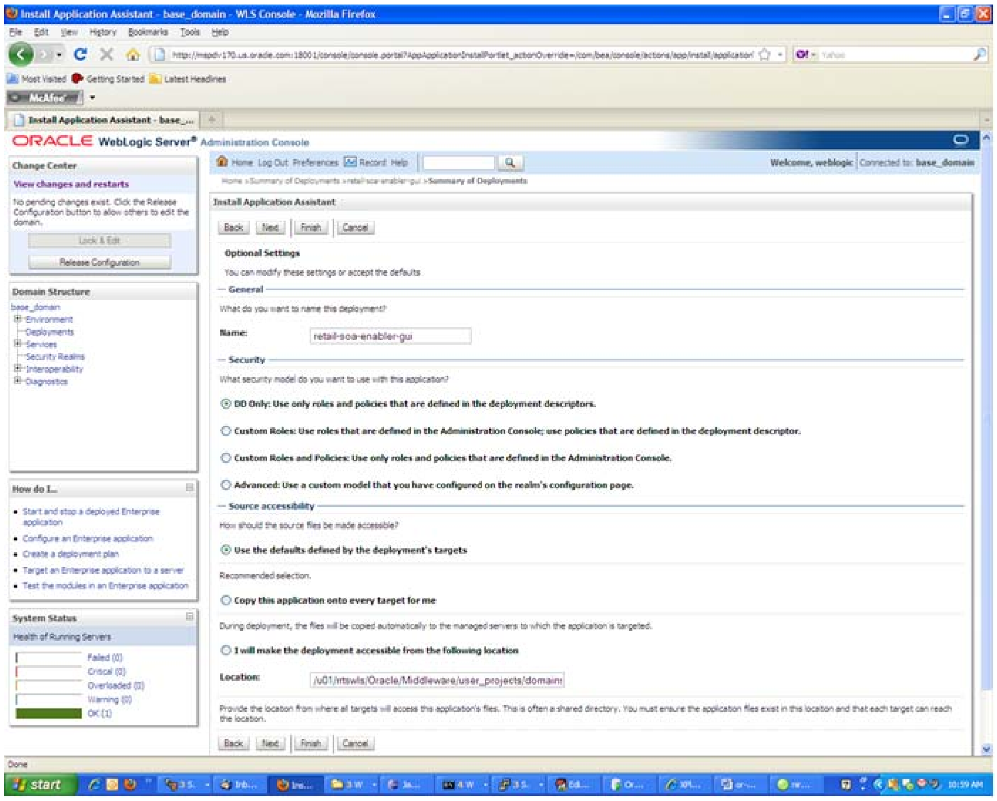
Select deployment targets

Select the servers and/or clusters to which you want to deploy this application. (You can reconfigure deployment targets later).

Available targets for retail-soa-enabler-gui :

Servers
✓ AdminServer
☐ igs-server-instance
☐ javaee-stubby-instance
☐ mapping-reports-instance
☐ rib-aip-wls-instance
☐ rib-func-artifact-wls-instance
☐ rib-rfm-wls-instance
☐ rib-rms-wls-instance
☐ rib-rms-wls-instance-1
☐ rib-rpm-wls-instance
☐ rib-rwms-wls-instance
☐ rib-sim-wls-instance
☐ rib-tafr-wls-instance
☐ riha-instance

8. Select Deployment Target. Click **Next** for Optional Settings.



Install Application Assistant

Optional Settings

You can modify these settings or accept the defaults

General

What do you want to name this deployment?

Name: retail-soa-enabler-gui

Security

What security model do you want to use with this application?

[✓ **DD Only: Use only roles and policies that are defined in the deployment descriptors.**]

Custom Roles: Use roles that are defined in the Administration Console; use policies that are defined in the deployment descriptor.

Custom Roles and Policies: Use only roles and policies that are defined in the Administration Console.

Advanced: Use a custom model that you have configured on the realm's configuration page.

Source accessibility

How should the source files be made accessible?

[✓ **Use the defaults defined by the deployment's targets**]

Recommended selection.

Copy this application onto every target for me

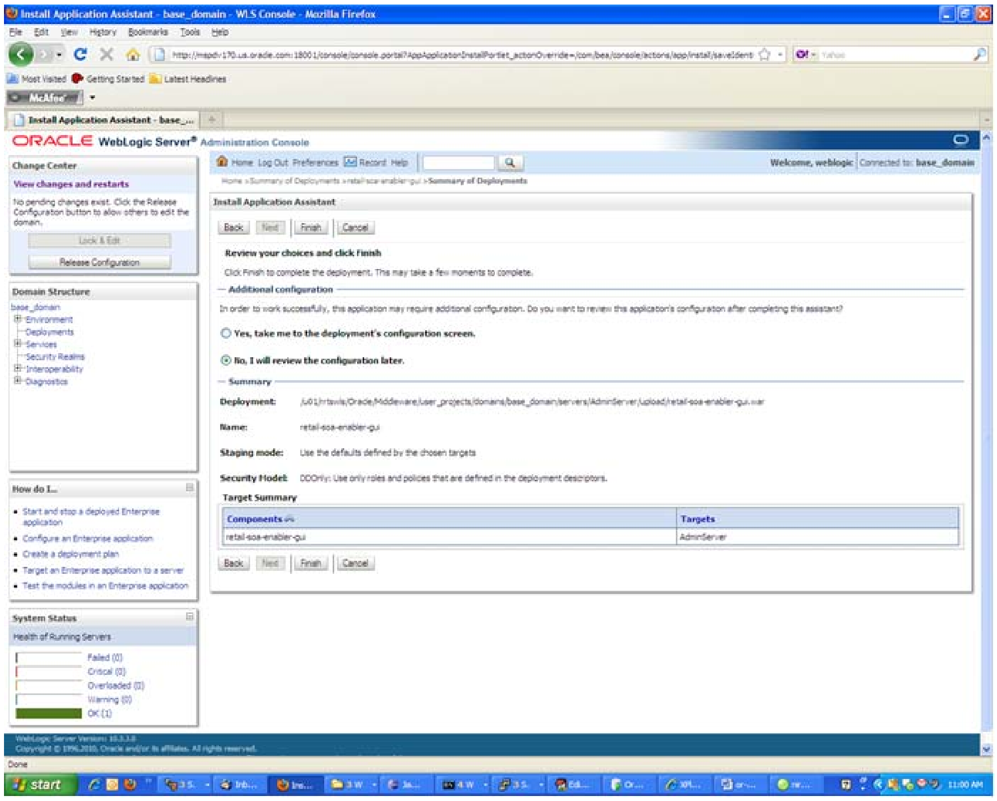
During deployment, the files will be copied automatically to the managed servers to which the application is targeted.

I will make the deployment accessible from the following location

Location: /u01/rtswls/Oracle/Middleware/user_projects/domain

Provide the location from where all targets will access this application's files. This is often a shared directory. You must ensure the application files exist in this location and that each target can reach the location.

9. Click **Next** to review your choices. Click **Finish**.



Install Application Assistant

Review your choices and click Finish

Click Finish to complete the deployment. This may take a few moments to complete.

Additional configuration

In order to work successfully, this application may require additional configuration. Do you want to review this application's configuration after completing this assistant?

Yes, take me to the deployment's configuration screen. [✓ No, I will review the configuration later.]

Summary

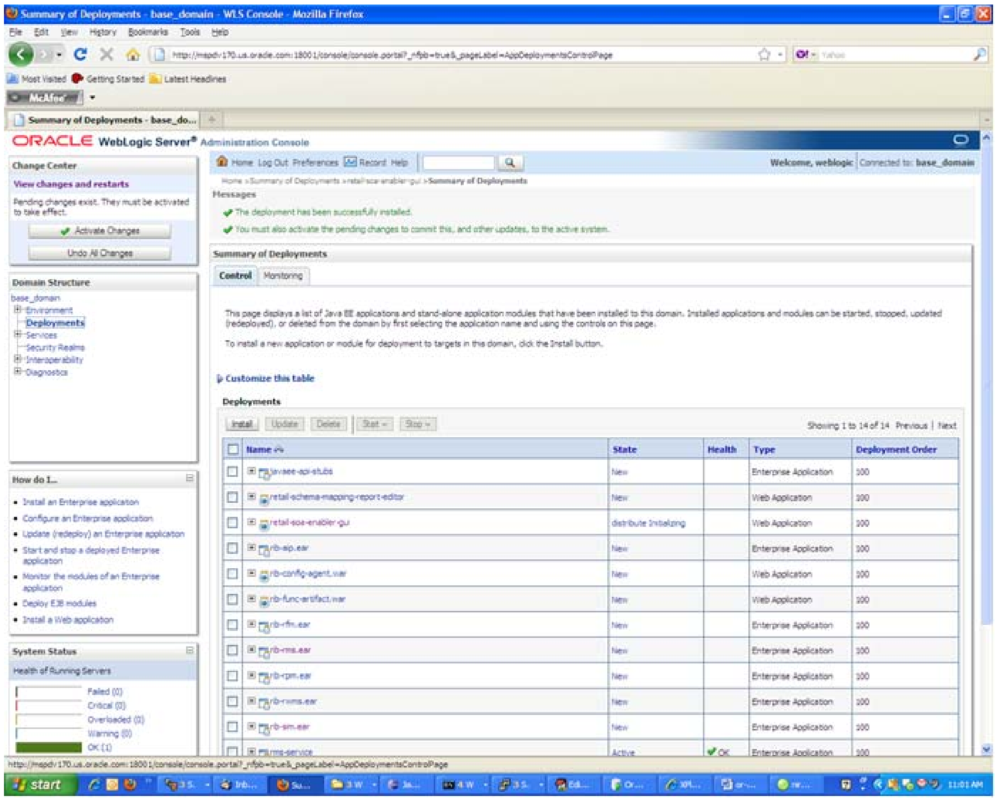
Deployment:	/u01/rtsvls/Oracle/Middleware/user_projects/domains/base_domain/servers/AdminServer/upload/retail-soa-enabler-gui.war
Name:	retail-soa-enabler-gui
Staging mode:	Use the defaults defined by the chosen targets
Security Model:	DDOnly: Use only roles and policies that are defined in the deployment descriptors.

Target Summary

Components	Targets
retail-soa-enabler-gui	AdminServer

10. Select **No, I will review the configuration later**.

11. Click **Finish** to deploy the application.



Summary of Deployments - base_domain - WLS Console - Mozilla Firefox

http://mspdv170.us.oracle.com:18001/console/console.portal?_nfpb=true&_pageLabel=AppDeploymentsControlPage

ORACLE WebLogic Server® Administration Console

Welcome, weblogic | Connected to: base_domain

Home >Summary of Deployments >retail-soa-enabler-gui >Summary of Deployments

Change Center

View changes and restarts

Pending changes exist. They must be activated to take effect.

Messages

✓ The deployment has been successfully installed.

✓ You must also activate the pending changes to commit this, and other updates, to the active system.

Summary of Deployments

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

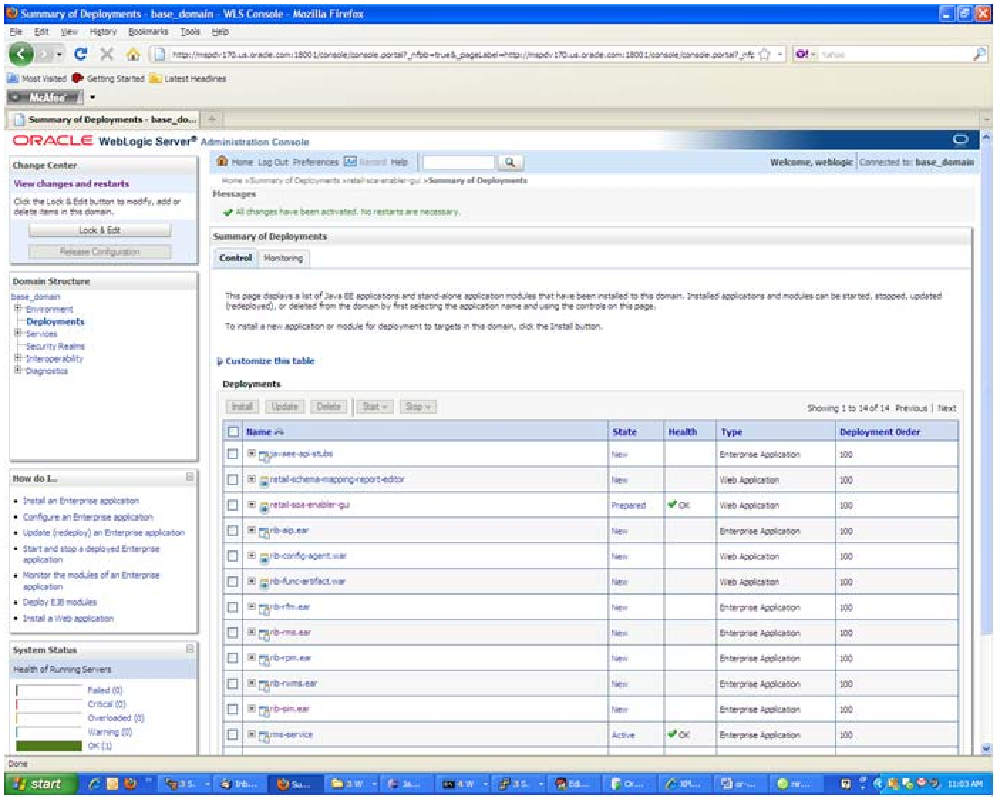
To install a new application or module for deployment to targets in this domain, click the Install button.

Deployments

Showing 1 to 14 of 14 Previous | Next

Name	State	Health	Type	Deployment Order
javaee-api-stubs	New		Enterprise Application	100
retail-schema-mapping-report-editor	New		Web Application	100
retail-soa-enabler-gui	distribute Initializing		Web Application	100
rib-aip.ear	New		Enterprise Application	100
rib-config-agent.war	New		Web Application	100
rib-func-artifact.war	New		Web Application	100
rib-rfm.ear	New		Enterprise Application	100
rib-rms.ear	New		Enterprise Application	100
rib-rpm.ear	New		Enterprise Application	100
rib-rwms.ear	New		Enterprise Application	100
rib-sim.ear	New		Enterprise Application	100
rms-service	Active	✓ OK	Enterprise Application	100

12. Click **Activate Changes** to finish the deployment.



Messages

All changes have been activated. No restarts are necessary.

Summary of Deployments

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

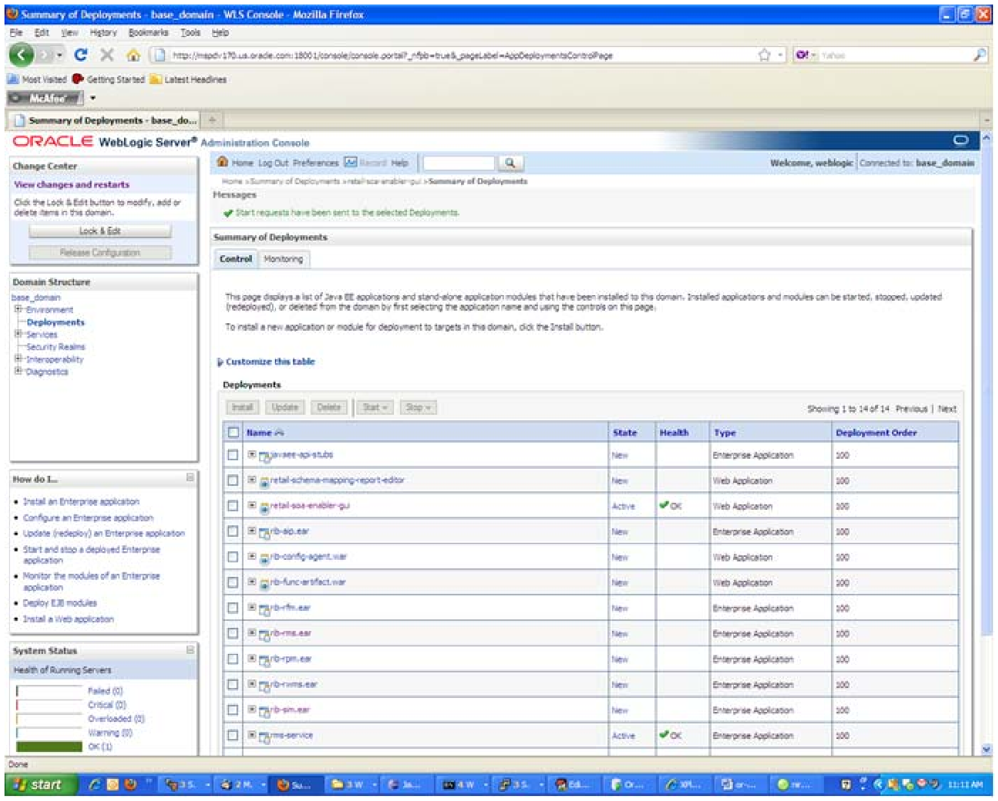
To install a new application or module for deployment to targets in this domain, click the Install button.

Deployments

Showing 1 to 14 of 14 Previous | Next

Name	State	Health	Type	Deployment Order
javaee-api-stubs	New		Enterprise Application	100
retail-schema-mapping-report-editor	New		Web Application	100
retail-soa-enabler-gui	Prepared	✓ OK	Web Application	100
rib-aip.ear	New		Enterprise Application	100
rib-config-agent.war	New		Web Application	100
rib-func-artifact.war	New		Web Application	100
rib-rfm.ear	New		Enterprise Application	100
rib-rms.ear	New		Enterprise Application	100
rib-rpm.ear	New		Enterprise Application	100
rib-rwms.ear	New		Enterprise Application	100
rib-sim.ear	New		Enterprise Application	100
rms-service	Active	✓ OK	Enterprise Application	100

13. Select the retail-soa-enabler-gui application. Click **Start > Servicing All Requests.**



Messages

Start requests have been sent to the selected Deployments.

Summary of Deployments

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

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

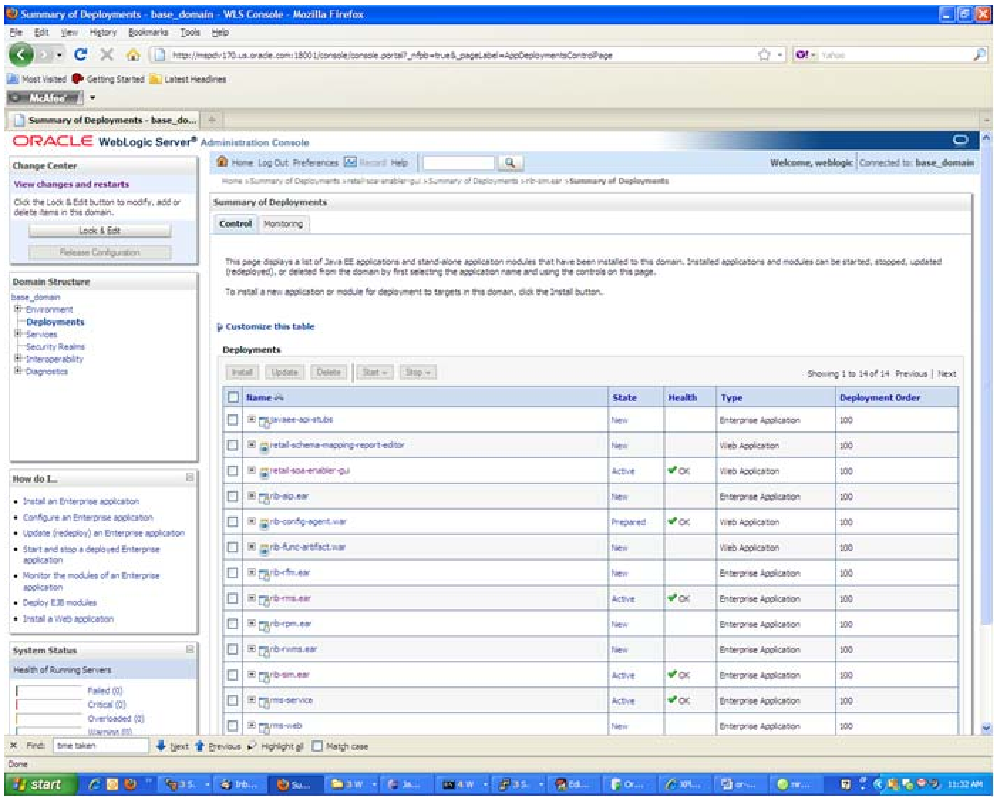
Deployments

Showing 1 to 14 of 14 Previous | Next

Name	State	Health	Type	Deployment Order
javaee-api-stubs	New		Enterprise Application	100
retail-schema-mapping-report-editor	New		Web Application	100
retail-soa-enabler-gui	Active	✓ OK	Web Application	100
rib-aip.ear	New		Enterprise Application	100
rib-config-agent.war	New		Web Application	100
rib-func-artifact.war	New		Web Application	100
rib-rfm.ear	New		Enterprise Application	100
rib-rms.ear	New		Enterprise Application	100
rib-rpm.ear	New		Enterprise Application	100
rib-rwms.ear	New		Enterprise Application	100
rib-sim.ear	New		Enterprise Application	100
rms-service	Active	✓ OK	Enterprise Application	100

Verify the Retail SOA Enabler Web Application

1. Navigate to the Deployments page.



Summary of Deployments - base_domain - WLS Console - Mozilla Firefox

http://mspdv170.us.oracle.com:18001/console/console.portal?_nfpb=true&_pageLabel=AppDeploymentsControlPage

Welcome, weblogic | Connected to: base_domain

Home >Summary of Deployments >retail-soa-enabler-gui >Summary of Deployments >rib-sim.ear >Summary of Deployments

Change Center

View changes and restarts

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

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

Health of Running Servers

Failed (0)
Critical (0)
Overloaded (0)
Warning (0)

Summary of Deployments

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

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

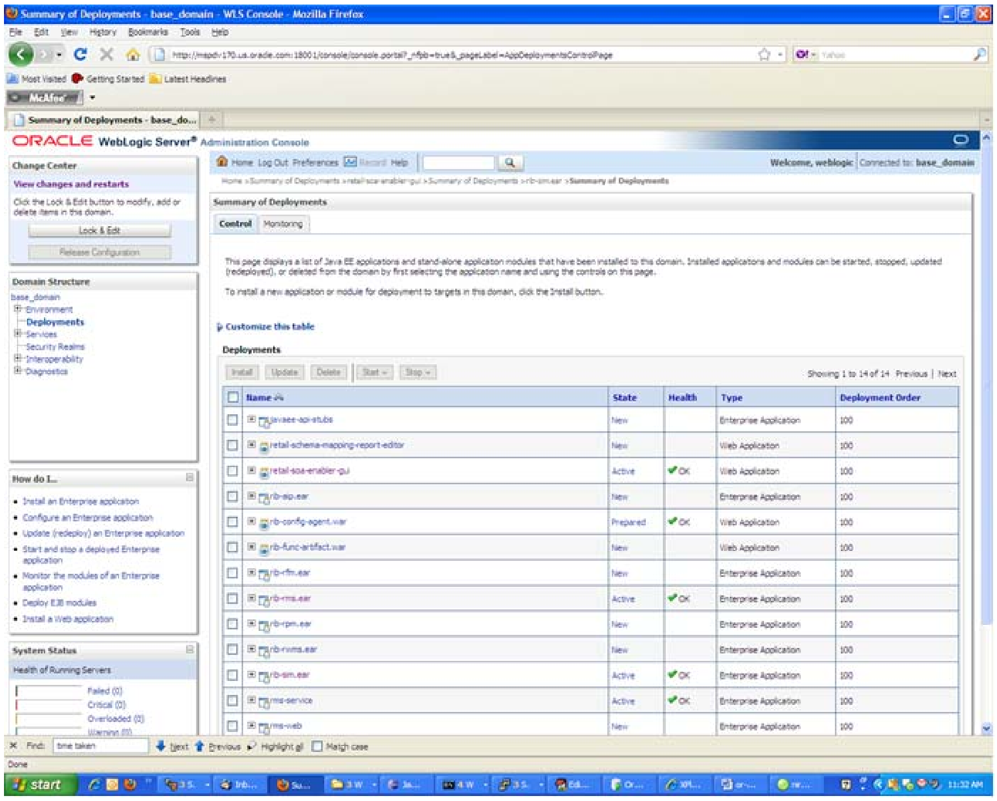
Customize this table

Deployments

Showing 1 to 14 of 14 Previous | Next

Name	State	Health	Type	Deployment Order
javaee-api-stubs	New		Enterprise Application	100
retail-schema-mapping-report-editor	New		Web Application	100
retail-soa-enabler-gui	Active	✓ OK	Web Application	100
rib-aip.ear	New		Enterprise Application	100
rib-config-agent.war	Prepared	✓ OK	Web Application	100
rib-func-artifact.war	New		Web Application	100
rib-rfm.ear	New		Enterprise Application	100
rib-rms.ear	Active	✓ OK	Enterprise Application	100
rib-rpm.ear	New		Enterprise Application	100
rib-rwms.ear	New		Enterprise Application	100
rib-sim.ear	Active	✓ OK	Enterprise Application	100
rms-service	Active	✓ OK	Enterprise Application	100
rms-web	New		Enterprise Application	100

2. On the Summary of Deployments screen, locate retail-soa-enabler-gui.



Summary of Deployments

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

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

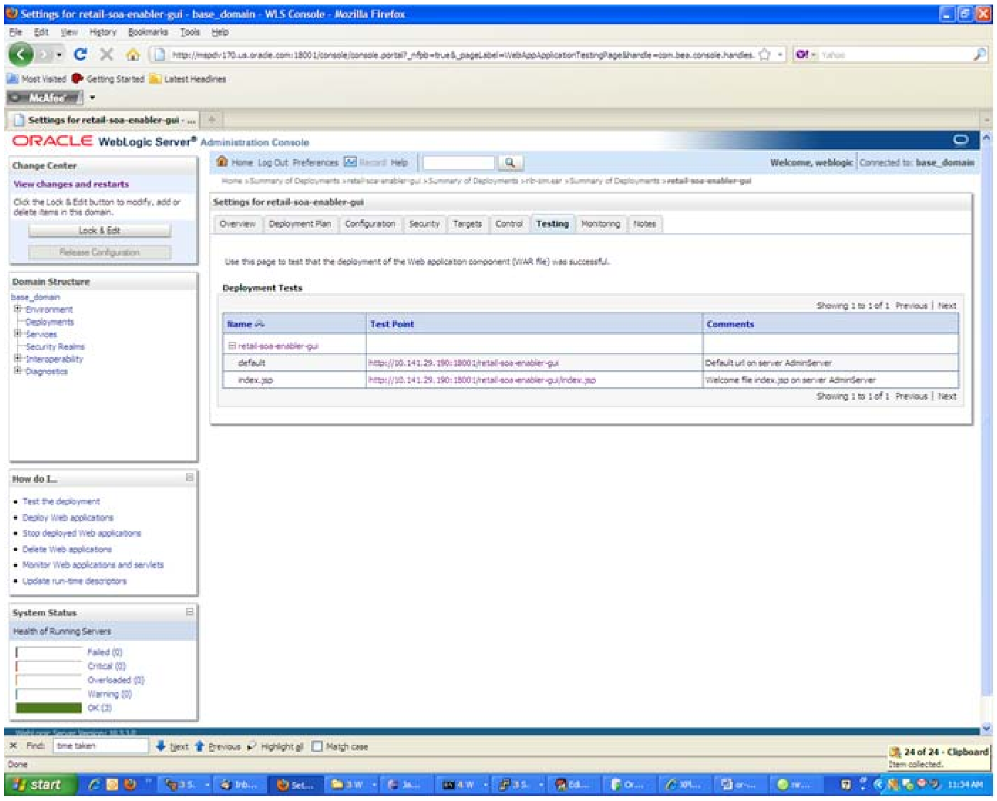
Deployments

Showing 1 to 14 of 14 Previous | Next

Name	State	Health	Type	Deployment Order
javaee-api-stubs	New		Enterprise Application	100
retail-schema-mapping-report-editor	New		Web Application	100
retail-soa-enabler-gui	Active	✓ OK	Web Application	100
rib-aip.ear	New		Enterprise Application	100
rib-config-agent.war	Prepared	✓ OK	Web Application	100
rib-func-artifact.war	New		Web Application	100
rib-rfm.ear	New		Enterprise Application	100
rib-rms.ear	Active	✓ OK	Enterprise Application	100
rib-rpm.ear	New		Enterprise Application	100
rib-rwms.ear	New		Enterprise Application	100
rib-sim.ear	Active	✓ OK	Enterprise Application	100
rms-service	Active	✓ OK	Enterprise Application	100
rms-web	New		Enterprise Application	100

3. Click **retail-soa-enabler-gui** to view settings for the retail-soa-enabler-gui.

4. Select the **Testing** tab.



Settings for retail-soa-enabler-gui

Use this page to test that the deployment of the Web application component (WAR file) was successful.

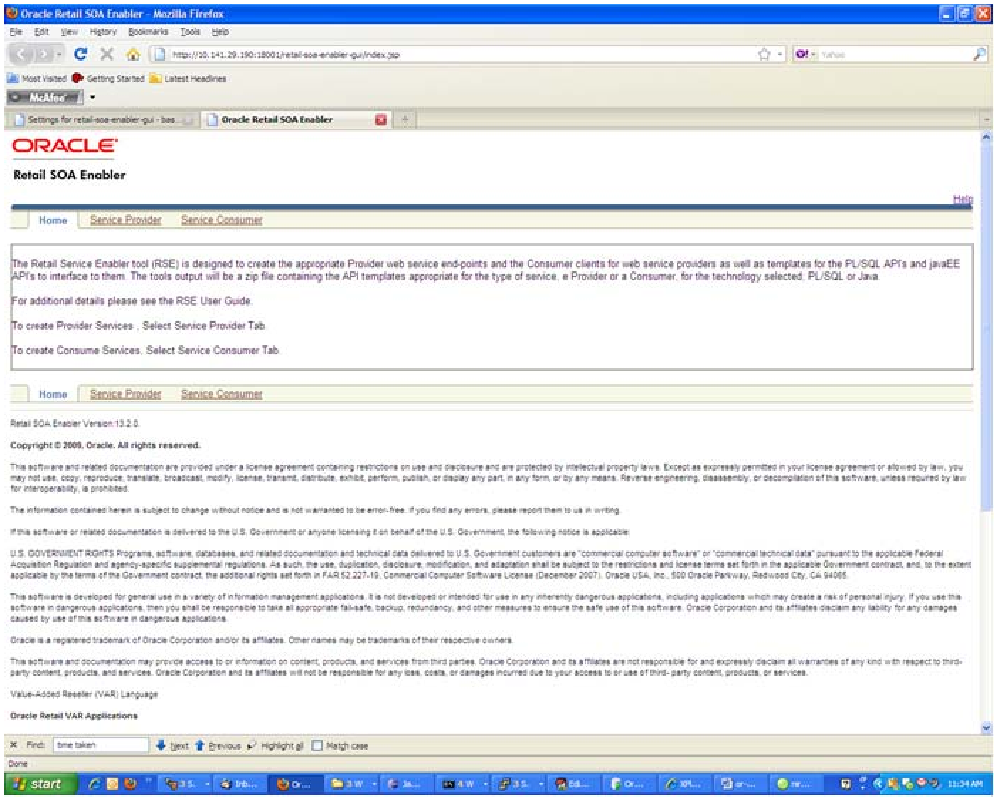
Deployment Tests

Showing 1 to 1 of 1 Previous | Next

Name	Test Point	Comments
retail-soa-enabler-gui		
default	http://10.141.29.190:18001/retail-soa-enabler-gui	Default url on server AdminServer
index.jsp	http://10.141.29.190:18001/retail-soa-enabler-gui/index.jsp	Welcome file index.jsp on server AdminServer

Showing 1 to 1 of 1 Previous | Next

5. Click the index.jsp URL in the Test Point.
6. The URL should open to the Retail Service-Oriented Architecture Enabler Home page.


Oracle Retail SOA Enabler - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://10.141.29.190:18001/retail-soa-enabler-gui/index.jsp
Most Visited Getting Started Latest Headlines
Settings for retail-soa-enabler-gui - bas...
Oracle Retail SOA Enabler
Retail SOA Enabler
Help
Home Service Provider Service Consumer
Find: time taken Next Previous Highlight all Match case
Done

The Retail Service Enabler tool (RSE) is designed to create the appropriate Provider web service end-points and the Consumer clients for web service providers as well as templates for the PL/SQL API's and javaEE API's to interface to them. The tools output will be a zip file containing the API templates appropriate for the type of service, e Provider or a Consumer, for the technology selected, PL/SQL or Java.

For additional details please see the RSE User Guide.

To create Provider Services , Select Service Provider Tab.

To create Consume Services, Select Service Consumer Tab.

Retail SOA Enabler Version 13.2.0.

Copyright © 2009, Oracle. All rights reserved.

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

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

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

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

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third- party content, products, or services.

Value-Added Reseller (VAR) Language

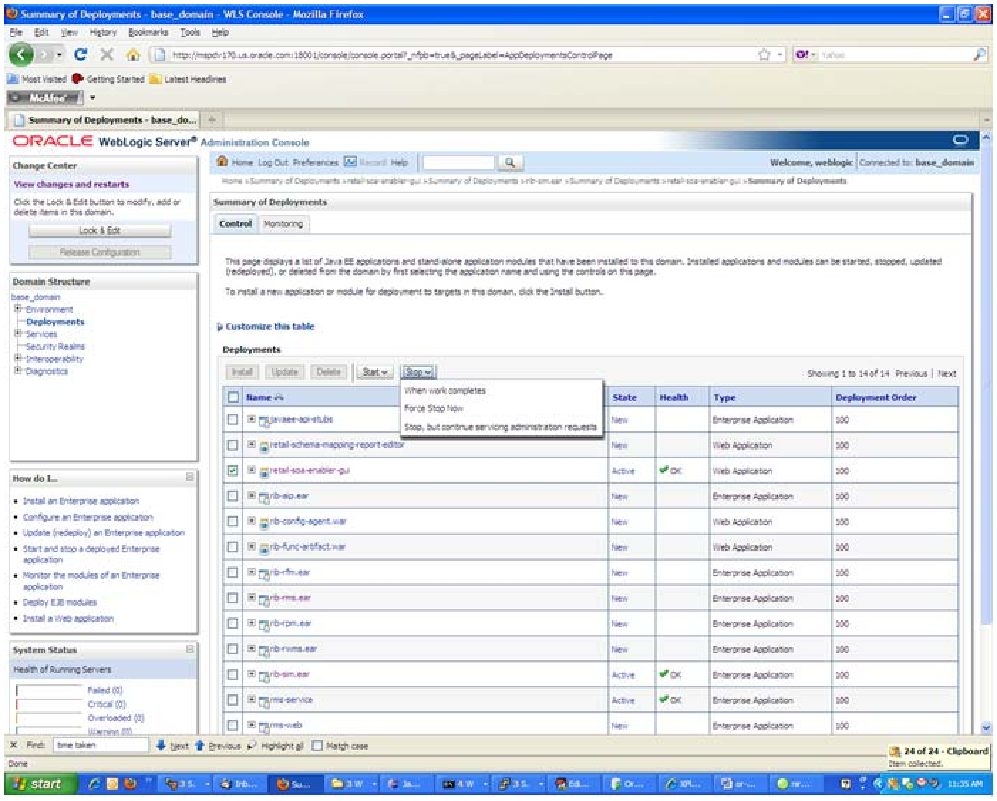
Oracle Retail VAR Applications

7. The installation is complete. See Chapter 4, "User Interface Usage."

Redeploy the Application

If the retail-soa-enabler-gui application has already been deployed, follow these steps:

1. If the retail-soa-enabler-gui application is running, select **Stop** and **When Work Completes** or **Force Stop Now**, depending on the environment. The recommended option always is **When Work Completes**.


Summary of Deployments - base_domain - WLS Console - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Most Visited Getting Started Latest Headlines
ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Change Center
View changes and restarts
Lock & Edit
Release Configuration
Domain Structure
base_domain
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
How do I...
System Status
Health of Running Servers
Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
Control
Monitoring
Customize this table
Install Update Delete Start Stop
When work completes
Force Stop Now
Stop, but continue servicing administration requests
Showing 1 to 14 of 14 Previous | Next

http://mspdv170.us.oracle.com:18001/console/console.portal?_nfpb=true&_pageLabel=AppDeploymentsControlPage

Home >Summary of Deployments >retail-soa-enabler-gui >Summary of Deployments >rib-sim.ear >Summary of Deployments >retail-soa-enabler-gui >**Summary of Deployments**

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

- 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

Summary of Deployments

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

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

Deployments

	Name	State	Health	Type	Deployment Order
☐	javaee-api-stubs	New		Enterprise Application	100
☐	retail-schema-mapping-report-editor	New		Web Application	100
✓	retail-soa-enabler-gui	Active	✓ OK	Web Application	100
☐	rib-aip.ear	New		Enterprise Application	100
☐	rib-config-agent.war	New		Web Application	100
☐	rib-func-artifact.war	New		Web Application	100
☐	rib-rfm.ear	New		Enterprise Application	100
☐	rib-rms.ear	New		Enterprise Application	100
☐	rib-rpm.ear	New		Enterprise Application	100
☐	rib-rwms.ear	New		Enterprise Application	100
☐	rib-sim.ear	Active	✓ OK	Enterprise Application	100
☐	rms-service	Active	✓ OK	Enterprise Application	100
☐	rms-web	New		Enterprise Application	100

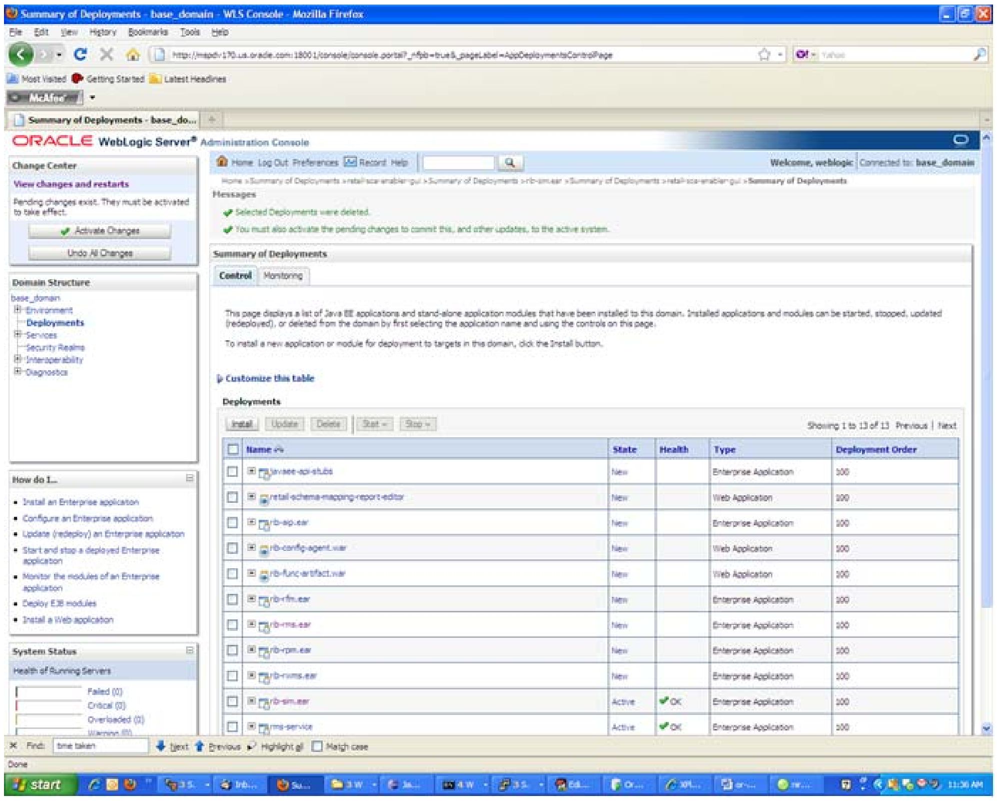
Find: time taken Next Previous Highlight all Match case

Done

24 of 24 - Clipboard
Item collected.

11:35 AM

2. Click **Lock & Edit.** Click **Delete**.


Summary of Deployments - base_domain - WLS Console - Mozilla Firefox

Change Center

View changes and restarts

Pending changes exist. They must be activated to take effect.

Home >Summary of Deployments >retail-sca-enabler-gui >Summary of Deployments >rb-sim.ear >Summary of Deployments >retail-sca-enabler-gui >Summary of Deployments

Messages

✓ Selected Deployments were deleted.

✓ You must also activate the pending changes to commit this, and other updates, to the active system.

Summary of Deployments

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

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

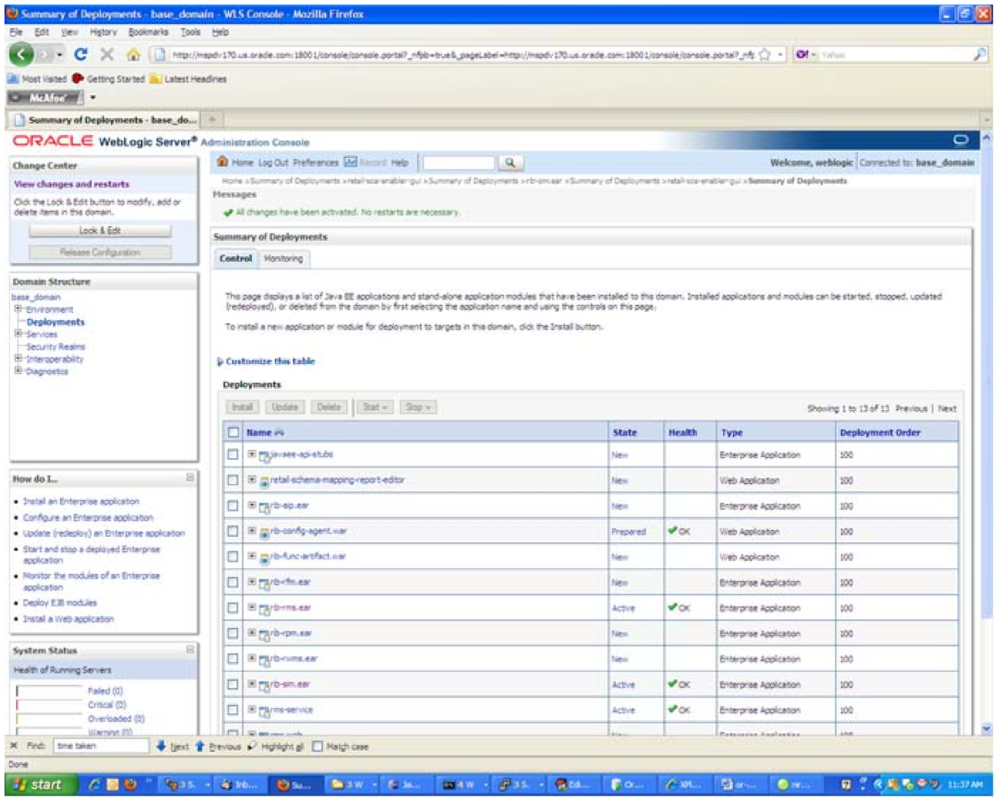
Deployments

Showing 1 to 13 of 13 Previous | Next

	Name	State	Health	Type	Deployment Order
☐	javaee-api-stubs	New		Enterprise Application	100
☐	retail-schema-mapping-report-editor	New		Web Application	100
☐	rb-aip.ear	New		Enterprise Application	100
☐	rb-config-agent.war	New		Web Application	100
☐	rb-func-artifact.war	New		Web Application	100
☐	rb-rfm.ear	New		Enterprise Application	100
☐	rb-rms.ear	New		Enterprise Application	100
☐	rb-rpm.ear	New		Enterprise Application	100
☐	rb-rwms.ear	New		Enterprise Application	100
☐	rb-sim.ear	Active	✓ OK	Enterprise Application	100
☐	rms-service	Active	✓ OK	Enterprise Application	100

Activate Changes
Undo All Changes
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
Health of Running Servers
Failed (0)
Critical (0)
Overloaded (0)
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Control Monitoring
Customize this table
Install Update Delete Start Stop

3. Click **Activate Changes.**



Summary of Deployments - base_domain - WLS Console - Mozilla Firefox

ORACLE WebLogic Server® Administration Console

Change Center

View changes and restarts

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

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

Health of Running Servers

Failed (0)
Critical (0)
Overloaded (0)

Messages

✓ All changes have been activated. No restarts are necessary.

Summary of Deployments

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

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

Customize this table

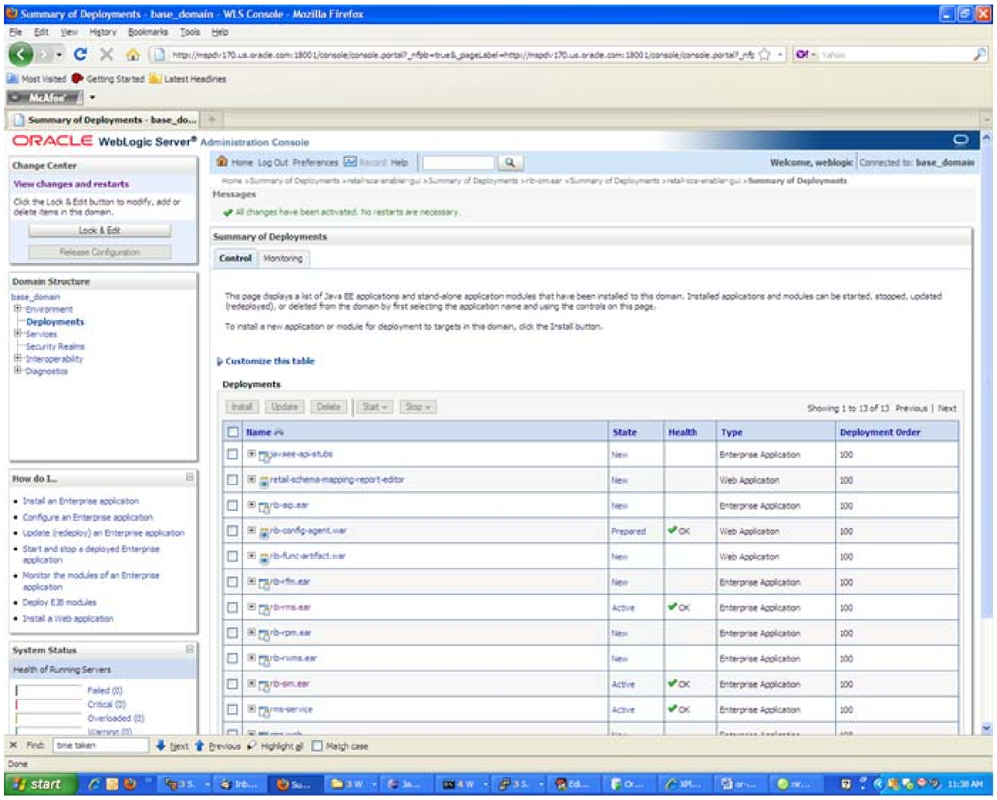
Deployments

Showing 1 to 13 of 13 Previous | Next

Name	State	Health	Type	Deployment Order
javaee-api-stubs	New		Enterprise Application	100
retail-schema-mapping-report-editor	New		Web Application	100
rb-aip.ear	New		Enterprise Application	100
rb-config-agent.war	Prepared	✓ OK	Web Application	100
rb-func-artifact.war	New		Web Application	100
rb-rfm.ear	New		Enterprise Application	100
rb-rms.ear	Active	✓ OK	Enterprise Application	100
rb-rpm.ear	New		Enterprise Application	100
rb-rwms.ear	New		Enterprise Application	100
rb-sim.ear	Active	✓ OK	Enterprise Application	100
rms-service	Active	✓ OK	Enterprise Application	100

Lock & Edit
Release Configuration
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Control
Monitoring
Install
Update
Delete
Start
Stop
Find: time taken
Next Previous Highlight all Match case
start
11:37 AM

4. The retail-soa-enabler-gui should now not show on the Summary of Deployment screen.



Home >Summary of Deployments >retail-soa-enabler-gui >Summary of Deployments >rib-sim.ear >Summary of Deployments >retail-soa-enabler-gui >Summary of Deployments

Messages

✓ All changes have been activated. No restarts are necessary.

Summary of Deployments

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

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

Deployments

Showing 1 to 13 of 13 Previous | Next

Name	State	Health	Type	Deployment Order
javaee-api-stubs	New		Enterprise Application	100
retail-schema-mapping-report-editor	New		Web Application	100
rib-aip.ear	New		Enterprise Application	100
rib-config-agent.war	Prepared	✓ OK	Web Application	100
rib-func-artifact.war	New		Web Application	100
rib-rfm.ear	New		Enterprise Application	100
rib-rms.ear	Active	✓ OK	Enterprise Application	100
rib-rpm.ear	New		Enterprise Application	100
rib-rwms.ear	New		Enterprise Application	100
rib-sim.ear	Active	✓ OK	Enterprise Application	100
rms-service	Active	✓ OK	Enterprise Application	100

B

Appendix: Sample ServiceProviderDefLibrary.xml

The sample below can be used as an initial template.

ServiceProviderDefLibrary.xml

```
<serviceProviderDefLibrary appName="rms"
xmlns="http://www.oracle.com/retail/integration/services/serviceProviderDefLibrary
/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

         <service name="Supplier"><!-- Noun, don't put suffix Service -->
                  <documentation />
                  <operation name="create"><!-- Verb -->
                           <documentation>Create a new
SupplierDesc.</documentation>
                           <input type="SupplierDesc"><!-- Existing BO -->
                                     <documentation>
                                            Input SupplierDesc to create.
                                     </documentation>
                           </input>
                           <output type="SupplierRef">
                                     <documentation>
                                             Return the SupplierRef for the newly
created
SupplierDesc.
                                     </documentation>
                           </output>
                           <fault faultType="IllegalArgumentWSFaultException">
                                     <documentation>
                                             Throw this exception when it is
"soap:Client" side
message problem.
                                    </documentation>
                           </fault>
                           <fault
                                 faultType="EntityAlreadyExistsWSFaultException">
                                   <documentation>
                                           Throw this exception when the object
already exist.
                                   </documentation>
                           </fault>
                           <fault faultType="IllegalStateWSFaultException">
                                   <documentation>
                                           Throw this exception when there is
```

```
unknown
"soap:Server" side problem.
                                   </documentation>
                           </fault>
                  </operation>
        <operation name="createSupSiteUsing"><!-- Verb -->
                               <documentation>Create a new
SupplierSite.</documentation>
                               <input type="SupplierDesc"><!-- Existing BO  -->
                                         <documentation>
                                                Input SupplierDesc to create.
                                         </documentation>
                               </input>
                               <output type="SupplierRef">
                                         <documentation>
                                                 Return the SupplierRef for the
newly created
SupplierDesc.
                                         </documentation>
                              </output>
                              <fault faultType="IllegalArgumentWSFaultException">
                                         <documentation>
                                                 Throw this exception when it is
"soap:Client" side
message problem.
                                         </documentation>
                              </fault>
                              <fault
                                faultType="EntityAlreadyExistsWSFaultException">
                                         <documentation>
                                                 Throw this exception when the
object already exist.
                                         </documentation>
                             </fault>
                             <fault faultType="IllegalStateWSFaultException">
                                         <documentation>
                                                 Throw this exception when there
is unknown
"soap:Server" side problem.
                                         </documentation>
                             </fault>
                   </operation>
          <operation name="createSupSiteAddrUsing"><!-- Verb -->
                            <documentation>Create a new
SupplierSite.</documentation>
                            <input type="SupplierDesc"><!-- Existing BO -->
                                         <documentation>
                                                 Input SupplierDesc to create.
                                         </documentation>
                            </input>
                            <output type="SupplierRef">
                                         <documentation>
                                                 Return the SupplierRef for the
newly created
SupplierDesc.
                                         </documentation>
                            </output>
                            <fault faultType="IllegalArgumentWSFaultException">
                                         <documentation>
                                                 Throw this exception when it is
```

```
"soap:Client" side
message problem.
                                         </documentation>
                            </fault>
                            <fault
                                 faultType="EntityAlreadyExistsWSFaultException">
                                       <documentation>
                                                Throw this exception when the
object already exist.
                                       </documentation>
                            </fault>
                            <fault faultType="IllegalStateWSFaultException">
                                       <documentation>
                                               Throw this exception when there is
unknown
"soap:Server" side problem.
                                       </documentation>
                            </fault>
                  </operation>
                  <operation name="update">
                            <input type="SupplierDesc" />
                            <output type="SupplierDesc" />
                            <fault faultType="IllegalArgumentWSFaultException" />
                            <fault
                                       faultType="EntityNotFoundWSFaultException"
/>
                            <fault faultType="IllegalStateWSFaultException" />
                  </operation>
    <operation name="updateSupSiteUsing">
                            <input type="SupplierDesc" />
                            <output type="SupplierDesc" />
                            <fault faultType="IllegalArgumentWSFaultException" />
                            <fault
                                       faultType="EntityNotFoundWSFaultException"
/>
                            <fault faultType="IllegalStateWSFaultException" />
                  </operation>
    <operation name="updateSupSiteOrgUnitUsing">
                            <input type="SupplierDesc" />
                            <output type="SupplierDesc" />
                            <fault faultType="IllegalArgumentWSFaultException" />
                            <fault
                                       faultType="EntityNotFoundWSFaultException"
/>
                            <fault faultType="IllegalStateWSFaultException" />
                  </operation>
     <operation name="updateSupSiteAddrUsing">
                            <input type="SupplierDesc" />
                            <output type="SupplierDesc" />
                            <fault faultType="IllegalArgumentWSFaultException" />
                            <fault
                                      faultType="EntityNotFoundWSFaultException"
/>
                            <fault faultType="IllegalStateWSFaultException" />
                  </operation>
                  <operation name="find" suffix="outputType">
                            <input type="SupplierRef" />
                            <output type="SupplierDesc" />
                            <fault faultType="IllegalArgumentWSFaultException" />
                            <fault
```

```
                                    faultType="EntityNotFoundWSFaultException"
/>
                          <fault faultType="IllegalStateWSFaultException" />
                </operation>
                <operation name="delete">
                           <input type="SupplierRef" />
                           <output type="SupplierRef" />
                           <fault faultType="IllegalArgumentWSFaultException" />
                           <fault
                                    faultType="EntityNotFoundWSFaultException"
/>
                          <fault faultType="IllegalStateWSFaultException" />
                </operation>
                <operation name="create">

                          <input type="SupplierCollectionDesc" />
                          <output type="SupplierCollectionRef" />
                          <fault faultType="IllegalArgumentWSFaultException" />
                          <fault faultType="EntityAlreadyExistsWSFaultException"
/>
                          <fault faultType="IllegalStateWSFaultException" />

                </operation>
                <operation name="update">
                          <input type="SupplierCollectionDesc" />
                          <output type="SupplierCollectionDesc" />
                          <fault faultType="IllegalArgumentWSFaultException" />
                          <fault
                                     faultType="EntityNotFoundWSFaultException"
/>
                          <fault faultType="IllegalStateWSFaultException" />
                </operation>
                <operation name="find" suffix="outputType">
                          <input type="SupplierCollectionRef" />
                          <output type="SupplierCollectionDesc" />
                          <fault faultType="IllegalArgumentWSFaultException" />
                          <fault
                                     faultType="EntityNotFoundWSFaultException"
/>
                          <fault faultType="IllegalStateWSFaultException" />
                </operation>
                <operation name="delete">
                          <input type="SupplierCollectionRef" />
                          <output type="SupplierCollectionRef" />
                          <fault faultType="IllegalArgumentWSFaultException" />
                          <fault
                                     faultType="EntityNotFoundWSFaultException"
/>
                          <fault faultType="IllegalStateWSFaultException" />
                </operation>
         </service>
</serviceProviderDefLibrary>
```

C

Appendix: Creating a JDBC Data Source

This section describes the steps required to create a JDBC data source.

Procedure

To create a JDBC data source, complete the following steps.

1. Log in to the WebLogic administration console. Use this URL: http://<host>:<listen port>/console/login/LoginForm.jsp.
2. Navigate the domain structure tree to Services/JDBC/Data Sources.
3. Click **New** to create the new data source. Enter the following required information.
   - Name: Select any name for the data source,
   - JNDI name: This field must be set to jdbc/RetailWebServiceDs. The generated code for the service uses this JNDI name to look up the data source.
4. Select the transaction options for the data source. Click **Next**.
5. Enter the database name and user information for the data source. Click **Next.**
6. The connection information for the data source is displayed. Click **Test Configuration** to ensure the connection information is correct. If the information is correct, the following message is displayed: "Connect test succeeded."
7. Click **Next**. Select a server to which to deploy the data source. (This step is not required at this point in the procedure if you want to deploy the data source to a server at a later time.)
8. Click **Finish** to complete the data source setup. The data sources page is displayed, including the new data source.
9. Click the new data source to see the properties page. A default connection pool is created for the data source. Click the **Connection Pool** tab to view the connection pool properties.
10. The generated JDBC connection URL for the data source is displayed in the following format:

    jdbc:oracle:thin:@<hostname>:<port>:<sid>

    For example:

    jdbc:oracle:thin:@localhost:1521:orc

11. If the database is a RAC database, the URL should be in the following format:

    dbc:oracle:thin:@(DESCRIPTION =(ADDRESS_LIST =(ADDRESS = (PROTOCOL = TCP)(HOST = <host>)(PORT = <port>))(ADDRESS = (PROTOCOL = TCP)(HOST = <host>)(PORT = <port>))(LOAD_BALANCE = yes))(CONNECT_ DATA =(SERVICE_NAME = <sid>)))

    For example:

    jdbc:oracle:thin:@(DESCRIPTION =(ADDRESS_LIST =(ADDRESS = (PROTOCOL = TCP)(HOST = mspvip72)(PORT = 1521))(ADDRESS = (PROTOCOL = TCP)(HOST = mspvip73)(PORT = 1521))(LOAD_BALANCE = yes))(CONNECT_ DATA =(SERVICE_NAME = dvolr02)))

12. In the Configuration > Connection Pool tab of the data source, set the following properties.

    - Initial capacity: Set the value to 20 connections. This value should be increased or decreased based on the expected load on the server.
    - Maximum capacity: Set the value to 100 connections. This value should be increased or decreased based on the expected load on the server.
    - Capacity Increment: Set the value to 20 connections. This value should be increased or decreased based on the expected load on the server.
    - Inactive Connection Time-out: This property is available in the advanced section of the connection pool configuration. Set the value of this property to 60 seconds.
    - Remove Infected Connections Enabled: This check box must be unchecked.

13. Restart the WebLogic instance to reflect the data source changes.