

Oracle® Communications Billing and Revenue Management Web Services Manager



Release 15.0
F86215-02
September 2024

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Oracle Communications Billing and Revenue Management Web Services Manager, Release 15.0

F86215-02

Copyright © 2017, 2024, Oracle and/or its affiliates.

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 is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware 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 that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	v
Documentation Accessibility	v
Diversity and Inclusion	v

1 Using Web Services

About WSDL Files and BRM Opcodes	1-1
Configuring BRM to Use PCM_OP_BAL_GET_ECE_BALANCES	1-8
About Validating Input and Output XML Data	1-8
About Developing and Testing Client Applications with Web Services Manager	1-8
Example of Testing a Web Services Implementation Using a Client Application	1-9
Testing the Web Service	1-11
Using Sample Web Services Programs	1-14
Example of Reading an Account Object in BRM Using Web Services	1-14
Sample SOAP Request Input XML File	1-14
Sample SOAP Response Output XML File	1-15
About Data Masking in Web Services Responses	1-17

2 Installing Web Services Manager

Installing Web Services Manager	2-1
Supported Servers	2-1
Uninstalling Web Services Manager	2-2

3 Deploying Web Services Manager

Deploying and Launching Web Services Manager on WebLogic Server	3-1
Deploying and Launching Web Services Manager on Tomcat Server	3-3
Deploying and Launching infranetwebsvc.war	3-4
Deploying and Launching BrmWebServices.war	3-4

4	Configuring Web Services Manager	
	About Connecting Web Services Manager to the BRM System	4-1
	Connecting Web Services Manager to the BRM System	4-2
	Changing the Instance of BRM to which Web Services Manager Connects	4-4
	Configuring Security for Web Services Manager	4-6
	Configuring Security for Web Services Manager in WebLogic Server	4-6
	Configuring Authentication for WebLogic Server	4-6
	Configuring WebLogic Security Policy on BRM Web Services for JAX-WS in WebLogic Server	4-8
	Configuring Security for Web Services Manager in Tomcat Server	4-9
	Configuring Authentication for Web Services Manager for JAX-WS in Tomcat Server	4-9
	Enabling SSL in Tomcat Server	4-10
	Disabling the JarScanner Feature in Tomcat Server	4-10
	Configuring Java Logging for the Application Server	4-11
	Configuring Java Logging for WebLogic Server	4-11
	Specifying the Java Unified Logging (JUL) Mechanism	4-11
	Creating a Startup Class	4-11
5	Securing Web Services Manager with OAuth 2.0	
	About the OAuth 2.0 Authorization Framework	5-1
	Setting Up Web Services Manager with OAuth 2.0	5-1
	Creating an OAuth Identity Domain	5-2
	Creating a Resource Server	5-2
	Creating an OAuth Client	5-3
	Validating Your OAuth Setup	5-3
	Configuring Web Services Manager	5-4
	Enabling OAuth Validation for Web Services Manager	5-4
	Sending SOAP Requests to BRM Web Services	5-5
6	Customizing Web Services	
	Setting Up Web Services Manager to Support Custom Opcodes	6-1
	Creating a Custom Web Service	6-4
	Generating the Schema Files for Your System	6-6
	Generating the Schema for Your Opcodes	6-6
	Generating the Schema for an Existing Opcode	6-7
	Creating Opcode Specification Schema Files	6-7
	Specifying the XSL Rules to Create the Opcode Schema	6-8

Preface

This guide provides guidelines for installing and setting up Oracle Communications Billing and Revenue Management (BRM) Web Services Manager. The sample procedures use WebLogic Server, but you can apply the concepts to any standards-compliant, vendor-supported application server that supports web services.

Before reading this guide, you should be familiar with implementing web services using WebLogic Server. See your WebLogic Server documentation for more information.

Audience

This document is intended for systems integrators, system administrators, database administrators, and other individuals who are responsible for installing, configuring, and customizing Web services for BRM.

Documentation Accessibility

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

Access to Oracle Support

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

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Using Web Services

Learn how to use Oracle Communications Billing and Revenue Management (BRM) Web Services Manager, which enables BRM opcodes to be exposed through Web services. Web Services Manager supports SOAP Web services and is packaged as an integration pack.

Topics in this document:

- [About WSDL Files and BRM Opcodes](#)
- [About Validating Input and Output XML Data](#)
- [About Developing and Testing Client Applications with Web Services Manager](#)
- [Testing the Web Service](#)
- [Using Sample Web Services Programs](#)
- [Example of Reading an Account Object in BRM Using Web Services](#)
- [About Data Masking in Web Services Responses](#)

About WSDL Files and BRM Opcodes

Web Services Manager exposes BRM opcodes as operations through different Web services. The Web services define the opcodes that can be called and the attributes to include.

The Web service APIs are grouped by functional area. For example, the **BRMBillServices** Web service defines the billing Web service APIs, and the **BRMPymtServices** Web service defines the payment Web service APIs. Web Services Manager includes one WSDL file for each Web service.

Web Services Manager contains two different types of WSDL files. One type is for Web services that support the payload as an XML string data type. The second type is for Web services that support the payload as an XML element data type. For example:

- The **BRMBalService** Web service defines balances Web service APIs that take the payload as an XML string data type.
- The **BRMBalService_v2** Web service defines balances Web service APIs that take the payload as an XML element data type.

File names with a **_v2** suffix support the payload as an XML element data type.

Note:

The WSDL and schema (XSD) files for Web services that support the payload as an XML string data type are included in the **infranetwebsvc.war** file. If you customize any Web services, copy the customized schema files and WSDL files to the **infranetwebsvc.war** file.

Web services that support the payload as an XML element data type describe the input in a well-defined structure. Any standards-compliant SOAP development application can generate a client stub.

Table 1-1 describes the Web services included that take the payload as an XML string.

Table 1-1 Web Services Included in Web Services Manager that Take the Payload as an XML String


Web Service Name	Description
BRMARServices	<p>Defines the accounts receivable Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_AR_ACCOUNT_ADJUSTMENT • PCM_OP_AR_BILL_ADJUSTMENT • PCM_OP_AR_GET_ACCT_ACTION_ITEMS • PCM_OP_AR_GET_ACCT_BAL_SUMMARY • PCM_OP_AR_GET_ACCT_BILLS • PCM_OP_AR_GET_BAL_SUMMARY • PCM_OP_AR_GET_BILL_ITEMS • PCM_OP_AR_ITEM_ADJUSTMENT • PCM_OP_AR_EVENT_ADJUSTMENT • PCM_OP_AR_GET_ACTION_ITEMS • PCM_OP_AR_GET_BILLS • PCM_OP_AR_RESOURCE_AGGREGATION <p>See "Accounts Receivable FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>
BRMBalServices	<p>Defines the balances Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_BAL_GET_BALANCES • PCM_OP_BAL_GET_BAL_GRP_AND_SVC • PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC • PCM_OP_BAL_GET_ACCT_BILLINFO • PCM_OP_BAL_GET_ECE_BALANCES <div style="border: 1px solid #0070C0; padding: 10px; margin-top: 10px;"> <p> Note:</p> <p>You must perform configuration steps before calling this opcode. See "Configuring BRM to Use PCM_OP_BAL_GET_ECE_BALANCES".</p> </div> <p>See "Balance FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>
BRMBillServices	<p>Defines the billing Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT • PCM_OP_BILL_GROUP_MOVE_MEMBER • PCM_OP_BILL_MAKE_BILL_NOW • PCM_OP_BILL_DEBIT • PCM_OP_BILL_GROUP_GET_PARENT <p>See "Billing FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>

Table 1-1 (Cont.) Web Services Included in Web Services Manager that Take the Payload as an XML String

Web Service Name	Description
BRMCollectionsServices	<p>Defines the collections Web service, which includes the following opcode:</p> <ul style="list-style-type: none"> • PCM_OP_COLLECTIONS_SET_ACTION_STATUS <p>See "Collections Manager FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>
BRMCustcareServices	<p>Defines the customer care Web service, which includes the following opcode:</p> <ul style="list-style-type: none"> • PCM_OP_CUSTCARE_MOVE_ACCT
BRMCustServices	<p>Defines the customer Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_CUST_COMMIT_CUSTOMER • PCM_OP_CUST_MODIFY_CUSTOMER • PCM_OP_CUST_UPDATE_CUSTOMER • PCM_OP_CUST_UPDATE_SERVICES • PCM_OP_CUST_DELETE_ACCT • PCM_OP_CUST_DELETE_PAYINFO • PCM_OP_CUST_CREATE_PROFILE • PCM_OP_CUST_MODIFY_PROFILE • PCM_OP_CUST_DELETE_PROFILE <p>See "Customer FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>
BRMInvServices	<p>Defines the invoicing Web service, which includes the following opcode:</p> <ul style="list-style-type: none"> • PCM_OP_INV_VIEW_INVOICE <p>Important: You must configure your client application to convert the invoice data received from the PCM_OP_INV_VIEW_INVOICE opcode into the appropriate format. See "About Invoicing Output XML Data" in <i>BRM JCA Resource Adapter</i>.</p> <p>See "Invoicing FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>
BRMPymtServices	<p>Defines the payment Web service, which includes the following opcode:</p> <ul style="list-style-type: none"> • PCM_OP_PYMT_COLLECT <p>See "Payment FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>
BRMReadServices	<p>Defines the read Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_READ_FLDS • PCM_OP_READ_OBJ • PCM_OP_SEARCH <p>See "LDAP Base Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>

Table 1-1 (Cont.) Web Services Included in Web Services Manager that Take the Payload as an XML String

Web Service Name	Description
BRMSubscriptionServices	<p>Defines the subscription Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT • PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT • PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION • PCM_OP_SUBSCRIPTION_CHANGE_DEAL • PCM_OP_SUBSCRIPTION_PURCHASE_DEAL • PCM_OP_SUBSCRIPTION_SET_BUNDLE • PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS • PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO • PCM_OP_SUBSCRIPTION_SET_PRODINFO • PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS • PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION • PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS <p>See "Subscription Management FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>

Table 1-2 describes the Web services that take the payload as an XML element.

Table 1-2 Web Services Included in Web Services Manager that Take the Payload as an XML Element

Web Service Name	Description
BRMACTServices_v2	<p>Defines the activity Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_ACT_FIND • PCM_OP_ACT_LOAD_SESSION <p>See "Activity FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>

Table 1-2 (Cont.) Web Services Included in Web Services Manager that Take the Payload as an XML Element


Web Service Name	Description
BRMARServices_v2	<p>Defines the accounts receivable Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_AR_ACCOUNT_ADJUSTMENT • PCM_OP_AR_ACCOUNT_WRITEOFF • PCM_OP_AR_BILL_ADJUSTMENT • PCM_OP_AR_BILL_DISPUTE • PCM_OP_AR_BILL_SETTLEMENT • PCM_OP_AR_BILL_WRITEOFF • PCM_OP_AR_BILLINFO_WRITEOFF • PCM_OP_AR_EVENT_ADJUSTMENT • PCM_OP_AR_EVENT_DISPUTE • PCM_OP_AR_EVENT_SETTLEMENT • PCM_OP_AR_GET_ACCT_ACTION_ITEMS • PCM_OP_AR_GET_ACCT_BAL_SUMMARY • PCM_OP_AR_GET_ACCT_BILLS • PCM_OP_AR_GET_ACTION_ITEMS • PCM_OP_AR_GET_BAL_SUMMARY • PCM_OP_AR_GET_BILLS • PCM_OP_AR_GET_BILL_ITEMS • PCM_OP_AR_GET_DISPUTES • PCM_OP_AR_GET_DISPUTE_DETAILS • PCM_OP_AR_GET_ITEMS • PCM_OP_AR_GET_ITEM_DETAILS • PCM_OP_AR_ITEM_ADJUSTMENT • PCM_OP_AR_ITEM_DISPUTE • PCM_OP_AR_ITEM_SETTLEMENT • PCM_OP_AR_ITEM_WRITEOFF • PCM_OP_AR_RESOURCE_AGGREGATION <p>See "Accounts Receivable FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>
BRMBALServices_v2	<p>Defines the balances Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_BAL_CHANGE_VALIDITY • PCM_OP_BAL_GET_BALANCES • PCM_OP_BAL_GET_ECE_BALANCES <div style="border: 1px solid #0070C0; padding: 5px; margin: 10px 0;"> <p> Note:</p> <p>You must perform configuration steps before calling this opcode. See "Configuring BRM to Use PCM_OP_BAL_GET_ECE_BALANCES".</p> </div> <ul style="list-style-type: none"> • PCM_OP_BAL_GET_BAL_GRP_AND_SVC • PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC • PCM_OP_BAL_GET_ACCT_BILLINFO <p>See "Balance FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>

Table 1-2 (Cont.) Web Services Included in Web Services Manager that Take the Payload as an XML Element

Web Service Name	Description
BRMBILLServices_v2	<p>Defines the billing Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_BILL_DEBIT • PCM_OP_BILL_FIND • PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT • PCM_OP_BILL_GROUP_GET_PARENT • PCM_OP_BILL_GROUP_MOVE_MEMBER • PCM_OP_BILL_ITEM_EVENT_SEARCH • PCM_OP_BILL_ITEM_REFUND • PCM_OP_BILL_MAKE_BILL_NOW • PCM_OP_BILL_REVERSE • PCM_OP_BILL_SET_LIMIT_AND_CR • PCM_OP_BILL_VIEW_INVOICE <p>See "Billing FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>
BRMCOLLECTIONSServices_v2	<p>Defines the collections Web service, which includes the following opcode:</p> <ul style="list-style-type: none"> • PCM_OP_COLLECTIONS_SET_ACTION_STATUS <p>See "Collections Manager FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>
BRMCUSTCAREServices_v2	<p>Defines the customer care Web service, which includes the following opcode:</p> <ul style="list-style-type: none"> • PCM_OP_CUSTCARE_MOVE_ACCT
BRMCUSTServices_v2	<p>Defines the customer Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_CUST_COMMIT_CUSTOMER • PCM_OP_CUST_CREATE_PROFILE • PCM_OP_CUST_DELETE_ACCT • PCM_OP_CUST_DELETE_PAYINFO • PCM_OP_CUST_DELETE_PROFILE • PCM_OP_CUST_FIND • PCM_OP_CUST_FIND_PAYINFO • PCM_OP_CUST_FIND_PROFILE • PCM_OP_CUST_GET_NOTE • PCM_OP_CUST_MODIFY_CUSTOMER • PCM_OP_CUST_MODIFY_PROFILE • PCM_OP_CUST_SET_NOTE • PCM_OP_CUST_SET_STATUS • PCM_OP_CUST_SET_TAXINFO • PCM_OP_CUST_UPDATE_CUSTOMER • PCM_OP_CUST_UPDATE_SERVICES <p>See "Customer FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p> <ul style="list-style-type: none"> • PCM_OP_CUST_POL_GET_PLANS • PCM_OP_CUST_POL_GET_DEALS • PCM_OP_CUST_POL_GET_PRODUCTS • PCM_OP_CUST_POL_READ_PLAN <p>See "Customer FM Policy Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>

Table 1-2 (Cont.) Web Services Included in Web Services Manager that Take the Payload as an XML Element

Web Service Name	Description
BRMINVServices_v2	<p>Defines the invoicing Web service, which includes the following opcode:</p> <ul style="list-style-type: none"> • PCM_OP_INV_VIEW_INVOICE <p>Important: You must configure your client application to convert the invoice data received from the PCM_OP_INV_VIEW_INVOICE opcode into the appropriate format. See "About Invoicing Output XML Data" in <i>BRM JCA Resource Adapter</i>.</p> <p>See "Invoicing FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>
BRMPYMTServices_v2	<p>Defines the payment Web service, which includes the following opcode:</p> <ul style="list-style-type: none"> • PCM_OP_PYMT_COLLECT <p>See "Payment FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>
BRMREADServices_v2	<p>Defines the read Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_READ_FLDS • PCM_OP_READ_OBJ • PCM_OP_SEARCH • PCM_OP_TEST_LOOPBACK <p>See "LDAP Base Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>
BRMSUBSCRIPTIONServices_v2	<p>Defines the subscription Web service, which includes the following opcodes:</p> <ul style="list-style-type: none"> • PCM_OP_SUBSCRIPTION_CANCEL_DEAL • PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT • PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT • PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION • PCM_OP_SUBSCRIPTION_CHANGE_DEAL • PCM_OP_SUBSCRIPTION_GET_HISTORY • PCM_OP_SUBSCRIPTION_PURCHASE_DEAL • PCM_OP_SUBSCRIPTION_PURCHASE_FEES • PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS • PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER • PCM_OP_SUBSCRIPTION_SET_BUNDLE • PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS • PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO • PCM_OP_SUBSCRIPTION_SET_PRODINFO • PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS • PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION • PCM_OP_SUBSCRIPTION_TRANSITION_DEAL • PCM_OP_SUBSCRIPTION_TRANSITION_PLAN • PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS <p>See "Subscription Management FM Standard Opcodes" in <i>BRM Opcode Guide</i> for more information.</p>

Configuring BRM to Use PCM_OP_BAL_GET_ECE_BALANCES

Before you can call the PCM_OP_BAL_GET_ECE_BALANCES opcode, you must configure BRM to support the opcode. To do so, add the following entry to your Connection Manager (CM) configuration file (*BRM_home/sys/cm/pin.conf*):

```
- cm em_group ece PCM_OP_BAL_GET_ECE_BALANCES
```

Stop and restart the CM for the changes to take effect.

About Validating Input and Output XML Data

Web Services Manager validates the input and output XML by comparing the XML fields and values against the opcode XML schema.

The opcode specifications, schemas, and WSDL files are packaged along with Web Services Manager. The package includes the **opspec.xsd** file and the **pin_opspec_to_schema** utility. Use the **opspec.xsd** file to write opcode specifications for custom opcodes that need to be exposed as a Web service. Use the **pin_opspec_to_schema** utility to generate the schema files from the opcode specification files.

To configure Web Services Manager to validate the input and output XML against the target opcode XML schema:

1. Open the *local_dir/WEB-INF/classes/Infranet.properties* file.
2. Add the following entries to the file:
 - **webservices.input.validation.enabled=true**
 - **webservices.output.validation.enabled=true**
3. Do one of the following:
 - If you are using WebLogic Server, copy the schema files packaged as a part of Web Services Manager installation from the *BRM_home/deploy/web_services/schemas* directory to the *local_dir/common/lib* directory.
 - If you are using any supported server, copy the schema files from the *BRM_home/deploy/web_services/schemas* directory to the *local_dir/WEB-INF/classes* directory.

About Developing and Testing Client Applications with Web Services Manager

You can develop custom applications that interact with BRM through Web Services Manager. Use a SOAP development environment that supports importing WSDL files (for example, SoapUI) to develop and test your custom Web service applications. SOAP development applications may have minor differences in product configuration. Consult your SOAP development application documentation for configuration information.

In general, do the following to develop and test your Web services applications:

1. Download and install a SOAP development application.
2. Configure a new project in your SOAP development application.
3. Write a client application that communicates with Web services using the SOAP protocol.

4. Import the Web service definitions using the WSDL files. See "[About WSDL Files and BRM Opcodes](#)" for more information on accessing WSDL files.
5. Run the required commands to set up your application server environment.
6. Configure the properties of the Web services operations in your SOAP development environment with valid credentials.
7. Send a Web service request to BRM from the SOAP development environment client.
8. View the Web service response in the SOAP development environment.

Example of Testing a Web Services Implementation Using a Client Application

To test your Web services implementation, write a client application that communicates with the Web service using the SOAP protocol.

This sample procedure demonstrates how to use the **TestClient.java** sample code with the PCM_OP_TEST_LOOPBACK opcode to verify communication between BRM and the Web service. The sample uses WebLogic Server, but you can apply the concepts to any other supported application server.

1. Do one of the following, which sets up the WebLogic Server environment:
 - If WebLogic is installed on a Linux host, run *WebLogic_home/wlserver/server/bin/setWLSEnv.sh*
 - If WebLogic is installed on a Windows host, run *WebLogic_home/server/bin/setenv.exe*

where *WebLogic_home* is the directory in which you installed the WebLogic Server.

2. Create an XML file (*some_name_1.xml*) using the following text:

```
<project name="buildWebservice" default="all">
<property name="buildDir" value="./myapps" />
<property name="jarFiles" value="jars" />
<target name="all" depends="jar" description="builds everything">
</target>
<target name="generate-client">
<clientgen wsdl="http://198.51.100.1:7001/infranetwebsvc/services/Infranet?
wsdl"
packageName="test_client"
destDir= "./myapps"/>
</target>
<target name="compile" depends="generate-client" description="compile
source files">
    <echo> Compiling adapter files</echo>
    <javac destdir="${buildDir}">
        <src path="${buildDir}"/>
    </javac>
</target>
<target name="jar" depends="compile" description="generate jar file(s)">
    <jar jarfile="clientStub.jar" basedir="${buildDir}">
        <exclude name = "**/*.java"/>
    </jar>
</target>
```

```
<target name="clean" description="remove files created by target prepare">
  <delete dir="${buildDir}"/>
</target>
</project>
```

This XML file uses the WebLogic Server clientgen task to automatically generate a utility library that provides low-level SOAP communication (client stubs).

3. Run the following command, which creates the client stubs:

```
ant -file some_name_1.xml
```

This process generates the **clientstubs.jar** file, which contains stubs used by the client. The test client code (*source_home*/**TestClient.java**, where *source_home* is the directory where your source code files are stored) then creates an flist, converts it to XML, and calls the PCM_OP_TEST_LOOPBACK opcode.

The following is a sample listing of **TestClient.java**:

```
import java.io.IOException;
import test_client.*; // corresponds to package name clientgen generated

public class TestClient {

    public static void main(String[] args) {
        try {

            String wsdlUrl =
"http://198.51.100.1:7001/infranetwebsvc/services/Infranet?wsdl";
            InfranetWebServiceService service = new InfranetServiceService_
Impl( wsdlUrl );
            InfranetWebService port = service.getInfranet();

            // convert flist to XML representation
            String XMLInput="<flist xmlns:xsi=\"http://www.w3.org/2001/
XMLSchema-instance\"><POID>0.0.0.1 /account 80408 8</POID></flist>";
            System.out.println("Input: " + XMLInput);
            // invoke web service 'opcode' method
            String result = port.opcode("TEST_LOOPBACK", XMLInput);

            System.out.println("result: "+ result);

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

4. Create another XML file (*some_name_2.xml*) using the following text:

 **Note:**

Replace the paths for the JAR files as required.

```
<project name="test_client" default="all">

<target name="all" depends="run"/>

<path id="classpath">
  <pathelement path="clientstubs.jar"/>
  <pathelement path=".\\classes"/>
  <pathelement path="D:\\WebServices\\webman\\lib\\jaxrpc.jar"/>
  <pathelement path="D:\\bea\\wlserver_10.3\\server\\lib\\wseeclient.jar"/>
  <pathelement path="D:\\bea\\wlserver_10.3\\server\\lib\\weblogic.jar"/>
</path>

c
<target name="compile">
  <mkdir dir="classes"/>
  <javac srcdir="src"
        destdir="classes"
        classpathref="classpath"
        />
</target>

<target name="run" depends="compile">
  <java classname="TestClient"
        fork="yes"
        classpathref="classpath">
  </java>
</target>

</project>
```

5. Build and run the test with the *some_name_2.xml* file using regular Ant tasks:

```
ant -file some_name_2.xml
```

Testing the Web Service

To test your Web services implementation, use a SOAP development application that supports importing WSDL files, for example SoapUI. SOAP development applications may have minor differences in project configuration. Consult your SOAP development environment documentation for configuration information.

To test your Web services implementation, write a client application that communicates with the Web service using the SOAP protocol.

 **Note:**

You can test Web services using SoapUI version 4.6.3.

The sample procedures use WebLogic Server, but you can apply the concepts to any other supported server.

For example, with WebLogic Server:

1. Run one of the following commands, which set up the WebLogic Server environment:
 - If WebLogic is installed on a Linux host: `WebLogic_home/wlserver/server/bin/setWLSEnv.sh`
 - If WebLogic is installed on a Windows host: `WebLogic_home/server/bin/setenv.exe` where `WebLogic_home` is the directory in which you installed the WebLogic Server.
2. Create an XML file (*some_name_1.xml*) using the following text:

```
<project name="buildWebservice" default="all">
<property name="buildDir" value="./myapps" />
<property name="jarFiles" value="jars" />
<target name="all" depends="jar" description="builds everything">
</target>
<target name="generate-client">
<clientgen wsdl="http://198.51.100.1:7001/infranetwebsvc/services/Infranet?
wsdl"
packageName="test_client"
destDir= "./myapps"/>
</target>
<target name="compile" depends="generate-client" description="compile source files">
  <echo> Compiling adapter files</echo>
  <javac destdir="${buildDir}">
    <src path="${buildDir}" />
  </javac>
</target>
<target name="jar" depends="compile" description="generate jar file(s)">
  <jar jarfile="clientStub.jar" basedir="${buildDir}">
    <exclude name = "**/*.java" />
  </jar>
</target>
<target name="clean" description="remove files created by target prepare">
  <delete dir="${buildDir}" />
</target>
</project>
```

This XML file uses the WebLogic Server `clientgen` task to automatically generate a utility library that provides low-level SOAP communication (client stubs).

3. Run the following command, which creates the client stubs:

```
ant -file some_name_1.xml
```

This process generates the `clientstubs.jar` file, which contains stubs used by the client. The test client code (`src\TestClient.java`) then creates an flist, converts it to XML, and calls the `PCM_OP_TEST_LOOPBACK` opcode.

The following is a sample listing of `TestClient.java`:

```

import java.io.IOException;
import test_client.*; // corresponds to package name clientgen generated

public class TestClient {

public static void main(String[] args) {
    try {

        String wsdlUrl =
"http://198.51.100.1:7001/infranetwebsvc/services/Infranet?wsdl";
        InfranetWebServiceService service = new InfranetServiceService_Impl( wsdl
Url );
        InfranetWebService port = service.getInfranet();

        // convert flist to XML representation
        String XMLInput="<flist xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-
instance\"><POID>0.0.0.1 /account 80408 8</POID></flist>";
        System.out.println("Input: " + XMLInput);
        // invoke web service 'opcode' method
        String result = port.opcode("TEST_LOOPBACK", XMLInput);

        System.out.println("result: "+ result);

    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
}

```

4. Create another XML file (*some_name_2.xml*) using the following text:

 **Note:**

Replace the paths for the JAR files as required.

```

<project name="test_client" default="all">

<target name="all" depends="run"/>

<path id="classpath">
    <pathelement path="clientstubs.jar"/>
    <pathelement path=".\\classes"/>
    <pathelement path="D:\\WebServices\\webman\\lib\\jaxrpc.jar"/>
    <pathelement path="D:\\bea\\wlserver_10.3\\server\\lib\\wseeclient.jar"/>
    <pathelement path="D:\\bea\\wlserver_10.3\\server\\lib\\weblogic.jar"/>
</path>

c
<target name="compile">
    <mkdir dir="classes"/>
    <javac srcdir="src"
        destdir="classes"
        classpathref="classpath"
    />
</target>

<target name="run" depends="compile">
    <java classname="TestClient"
        fork="yes"

```

```
        classpathref="classpath">
      </java>
    </target>

  </project>
```

5. Build and run the test with the *some_name_2.xml* file using regular Ant tasks:

```
ant -file some_name_2.xml
```

Using Sample Web Services Programs

Web Services Manager includes sample programs that demonstrate how to write code for various tasks when customizing Web services. For example, the sample program **InfranetBALTestClient.java** creates an flist, converts it to XML, and calls the PCM_OP_BAL_GET_BALANCES opcode.

Example of Reading an Account Object in BRM Using Web Services

This section describes an example of reading an account object using Web services. It shows the sample SOAP request and a response message.

To read an account object in BRM using Web services, you call the `pcmOpReadObj` Web service API that maps to the `PCM_OP_READ_OBJ` opcode. The `pcmOpReadObj` Web service API is included in the **BRMReadServices_v2** Web service, which contains Web service APIs that are related to read accounts. See "[About WSDL Files and BRM Opcodes](#)" for more information about the Web services included in the Web Services Manager package.

You use URLs to create SOAP clients for Web services. The URL is generated by the JAX-WS in Oracle WebLogic Server.

To generate the URL for a Web service:

1. Log in to the WebLogic Server Administration Console.
2. Go to the deployments section.
3. Click **BrmWebServices** link.

The Settings page for the deployment appears and all the web services are listed in the **Modules and Components** section.

4. Click the web service in the **Modules and Components** section.
5. Click the **Testing** tab.
6. The WebLogic Server Administration console displays a URL for the BRM web services.

A sample URL for the **BRMReadServices_v2** Web service is as follows:

```
http://hostIPAddress:port/BrmWebServices/BRMReadServices_v2?wsdl
```

To call a Web service, users are required to authenticate using a valid user name and a password. Users can call only those Web services that they are authorized to call.

Sample SOAP Request Input XML File

The following sample shows a SOAP request for the **pcmOpReadObj** Web service API:

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bus="http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes">
  <soapenv:Header/>
  <soapenv:Body>
    <bus:pcmOpReadObjRequest>
      <bus:PCM_OP_READ_OBJ_Request>
        <bus:flags>0</bus:flags>
        <bus:PCM_OP_READ_OBJ_inputFlist>
          <bus:POID>0.0.0.1 /account 1 0</bus:POID>
        </bus:PCM_OP_READ_OBJ_inputFlist>
      </bus:PCM_OP_READ_OBJ_Request>
    </bus:pcmOpReadObjRequest>
  </soapenv:Body>
</soapenv:Envelope>

```

Sample SOAP Response Output XML File

The following sample shows a SOAP response message for the **pcmOpReadObj** Web service API:

```

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <pcmOpReadObjResponse xmlns="http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes">
      <brm:AAC_ACCESS xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
      <brm:AAC_PACKAGE xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
      <brm:AAC_PROMO_CODE xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
      <brm:AAC_SERIAL_NUM xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
      <brm:AAC_SOURCE xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
      <brm:AAC_VENDOR xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
      <brm:ACCESS_CODE1 xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
      <brm:ACCESS_CODE2 xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
      <brm:ACCOUNT_NO xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">ROOT</brm:ACCOUNT_NO>
      <brm:ACCOUNT_TAG xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
      <brm:ACCOUNT_TYPE xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">2</brm:ACCOUNT_TYPE>
      <brm:ATTRIBUTE xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">0</brm:ATTRIBUTE>
      <brm:BAL_GRP_OBJ xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">0.0.0.1 /balance_group 1 0</brm:BAL_GRP_OBJ>
      <brm:BRAND_OBJ xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">0.0.0.1 /account 1 0</brm:BRAND_OBJ>
      <brm:BUSINESS_TYPE xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">0</brm:BUSINESS_TYPE>
      <brm:CLOSE_WHEN_T xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">1970-01-01T00:00:00Z</brm:CLOSE_WHEN_T>
      <brm:CONTEXT_INFO xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">
        <brm:CORRELATION_ID>1724742721778T49</brm:CORRELATION_ID>
      </brm:CONTEXT_INFO>
      <brm:CREATED_T xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">2024-05-05T15:26:19Z</brm:CREATED_T>
      <brm:CURRENCY xmlns:brm="http://xmlns.oracle.com/BRM/schemas/

```

```

BusinessOpcodes">840</brm:CURRENCY>
  <brm:CURRENCY_SECONDARY xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">0</brm:CURRENCY_SECONDARY>
  <brm:CUSTOMER_SEGMENT_LIST xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
  <brm:EFFECTIVE_T xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">2024-05-05T15:26:19Z</brm:EFFECTIVE_T>
  <brm:GL_SEGMENT xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
  <brm:GROUP_OBJ xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">0.0.0.0 0 0</brm:GROUP_OBJ>
  <brm:INCORPORATED_FLAG xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">0</brm:INCORPORATED_FLAG>
  <brm:INTERNAL_NOTES flags="0x00" offset="0" size="0" xmlns:brm="http://
xmlns.oracle.com/BRM/schemas/BusinessOpcodes"/>
  <brm:ITEM_POID_LIST xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">0.0.0.1|/item/misc 1 0</brm:ITEM_POID_LIST>
  <brm:LASTSTAT_CMNT xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
  <brm:LAST_STATUS_T xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">2024-05-05T15:26:19Z</brm:LAST_STATUS_T>
  <brm:LINEAGE xmlns:brm="http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes"/></
brm:LINEAGE>
  <brm:LOCALE xmlns:brm="http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes"/>
  <brm:MOD_T xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">2024-05-05T15:26:19Z</brm:MOD_T>
  <brm:NAME xmlns:brm="http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes">Brand
Host</brm:NAME>
  <brm:NAMEINFO elem="1" xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">
    <brm:ADDRESS/>
    <brm:CANON_COMPANY/>
    <brm:CANON_COUNTRY/>
    <brm:CITY/>
    <brm:COMPANY/>
    <brm:CONTACT_TYPE/>
    <brm:COUNTRY/>
    <brm:EMAIL_ADDR/>
    <brm:FIRST_CANON>system</brm:FIRST_CANON>
    <brm:FIRST_NAME>System</brm:FIRST_NAME>
    <brm:GEOCODE/>
    <brm:LAST_CANON>administrator</brm:LAST_CANON>
    <brm:LAST_NAME>Administrator</brm:LAST_NAME>
    <brm:MIDDLE_CANON/>
    <brm:MIDDLE_NAME/>
    <brm:SALUTATION/>
    <brm:SERVICE_OBJ>0.0.0.0 0 0</brm:SERVICE_OBJ>
    <brm:STATE/>
    <brm:TAXPKG_TYPE>0</brm:TAXPKG_TYPE>
    <brm:TITLE/>
    <brm:ZIP/>
  </brm:NAMEINFO>
  <brm:NEXT_ITEM_POID_LIST xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes"/>
  <brm:OBJECT_CACHE_TYPE xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">0</brm:OBJECT_CACHE_TYPE>
  <brm:POID xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">0.0.0.1 /account 1 1</brm:POID>
  <brm:READ_ACCESS xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">L</brm:READ_ACCESS>
  <brm:RESIDENCE_FLAG xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">0</brm:RESIDENCE_FLAG>

```

```
<brm:STATUS xmlns:brm="http://xmlns.oracle.com/BRM/schemas/  
BusinessOpcodes">10100</brm:STATUS>  
<brm:STATUS_FLAGS xmlns:brm="http://xmlns.oracle.com/BRM/schemas/  
BusinessOpcodes">0</brm:STATUS_FLAGS>  
<brm:TIMEZONE_ID xmlns:brm="http://xmlns.oracle.com/BRM/schemas/  
BusinessOpcodes"/>  
<brm:VAT_CERT xmlns:brm="http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes"/>  
<brm:WRITE_ACCESS xmlns:brm="http://xmlns.oracle.com/BRM/schemas/  
BusinessOpcodes">L</brm:WRITE_ACCESS>  
</pcmOpReadObjResponse>  
</S:Body>  
</S:Envelope>
```

About Data Masking in Web Services Responses

SOAP output response XML files may contain masked fields as configured by your BRM implementation. Subscriber fields, including payment information and user credentials, may be hidden in responses for securing sensitive subscriber data.

See "Masking Sensitive Customer Data" in *BRM Managing Customers* for more information on configuring data masking.

2

Installing Web Services Manager

Learn how to install and set up Oracle Communications Billing and Revenue Management (BRM) Web Services Manager.

Topics in this document:

- [Installing Web Services Manager](#)
- [Uninstalling Web Services Manager](#)

Installing Web Services Manager

 **Note:**

If you already installed the product, you must uninstall its features before reinstalling them.

Before you install and configure Web Services Manager, you must install the following:

- A supported, standards-compliant server. See "[Supported Servers](#)" for a list of the servers supported by Web Services Manager. See server documentation for more information.
- BRM. See "Installing BRM" in *BRM Installation Guide* for more information.

You must also increase the heap size used by the Java Virtual Machine (JVM) before running the installation program to avoid "Out of Memory" error messages in the log file. For information, see "Problem: Java Out of Memory Error" in *BRM System Administrator's Guide*.

 **Note:**

Oracle recommends that you install Web Services Manager on the system on which BRM is installed.

To install Web Services Manager, see "Installing Individual BRM Components" in *BRM Installation Guide*.

Supported Servers

Web Services Manager is supported on the following servers:

- Oracle WebLogic Server
- Apache Tomcat server

Uninstalling Web Services Manager

To uninstall Web Services Manager, see "Uninstalling Optional Components" in *BRM Installation Guide*.

3

Deploying Web Services Manager

Learn how to deploy and launch the Oracle Communications Billing and Revenue Management (BRM) Web Services Manager.

Topics in this document:

- [Deploying and Launching Web Services Manager on WebLogic Server](#)
- [Deploying and Launching Web Services Manager on Tomcat Server](#)

Deploying and Launching Web Services Manager on WebLogic Server

You can deploy Web Services Manager on WebLogic Server through the WebLogic Server Administration Console. Depending on the type of payload supported by web services, deploy one of the following files:

- **infranetwebsvc.war**: Includes web services that support the payload as an XML string data type.
- **BrmWebServices.war**: Includes web services that support the payload as an XML element data type.

If you customize web services, regenerate **infranetwebsvc.war** or **BrmWebServices.war** and use the generated version. Otherwise, you should use the default **infranetwebsvc.war** or **BrmWebServices.war** file. For more information about customizing web services, see "[Customizing Web Services](#)".

To deploy Web Services Manager on WebLogic Server:

1. Create the WebLogic Server domain. See the discussion about creating a WebLogic domain in *Fusion Middleware Creating Domains Using the Configuration Wizard* for detailed instructions.
2. If you deploy the **BrmWebServices.war** file, set the heap size required to start WebLogic Server:
 - a. Open the `WebLogic_home/user_projects/domains/Domain_Name/setDomainEnv.sh` file in a text editor.

where `WebLogic_home` is the directory in which WebLogic Server is installed, and `domain_name` is the name of the domain you created in step 1.
 - b. Add the following entry:

```
USER_MEM_ARGS="-Xms2048m -Xmx2048m"
```
 - c. Save and close the file.
 - d. Restart WebLogic Server.
3. Do one of the following:
 - If you customized web services:

- a. Extract the `BRM_home/deploy/web_services/infranetwebsvc.war` or the `BRM_home/deploy/web_services/BrmWebServices.war` file to `local_dir`.
where `BRM_home` is the directory in which BRM is installed, and `local_dir` is a directory on the machine on which you installed WebLogic Server.
- b. Copy the `CustomFields.jar` files to the `local_dir/WEB-INF/lib` directory. See ["Setting Up Web Services Manager to Support Custom Opcodes"](#) for more information.

 **Note:**

The JRE version that was used to generate `CustomFields.jar` must be the same or lower than the version of the WebLogic Server JRE.

- c. Open the `BRM_home/deploy/web_services/Infranet.properties` file in a text editor.
- d. Modify the following entry:
`infranet.custom.field.package = package`
where `package` is the name of the package that contains the `CustomOp.java` file. For example: `com.portal.classFiles`.

- e. Add all the custom fields to the `Infranet.properties` file.
- f. Save and close the file.
- g. Copy the `BRM_home/deploy/web_services/Infranet.properties` file to the `local_dir/WEB-INF/classes` or in the home directory on the machine on which WebLogic Server is installed.

- h. Regenerate the WAR file by running one of the following commands:

To regenerate the `infranetwebsvc.war` file:

```
jar -cvf infranetwebsvc.war *
```

To regenerate the `BrmWebServices.war` file:

```
jar -cvf BrmWebServices.war *
```

- If you did not customize web services:

- a. Extract the `BRM_home/deploy/web_services/infranetwebsvc.war` or the `BRM_home/deploy/web_services/BrmWebServices.war` file to `local_dir`.
- b. Copy the `BRM_home/deploy/web_services/Infranet.properties` file to the `local_dir/WEB-INF/classes` directory or in the home directory on the machine on which WebLogic Server is installed.
- c. Regenerate the WAR file by running one of the following commands:

To regenerate the `infranetwebsvc.war` file:

```
jar -cvf infranetwebsvc.war *
```

To regenerate the `BrmWebServices.war` file:

```
jar -cvf BrmWebServices.war *
```

4. Log in to WebLogic Server Administration Console.
5. In the Domain Structure pane, click **Deployments**.

The Summary of Deployments page appears.

6. Click **Install**.

The Install Application Assistant page appears.

7. Enter the local directory path for **infranetwebsvc.war** or **BrmWebServices.war**, and then press **Enter**.
8. Select **Install this deployment as an application**, and then click **Next**.
9. Select the server or servers on which you want to deploy Web Services Manager, and then click **Next**.
10. Click **Next**.

 **Note:**

When you deploy Web Services Manager on WebLogic Server, select the **Custom Roles and Policies** option in the Security section of WebLogic Server Administration Console.

11. Verify your deployment options, and then click **Finish**.

WebLogic Server displays the deployed application on the Deployments page.

To launch Web Services Manager for web services:

1. From the WebLogic Server Administration Console Home page, select **Deployments** in the Domain Structure pane.

The Summary of Deployments page appears.

2. Click the **BrmWebServices** link.
A list of all web services is displayed.
3. Click on any of the web service.
4. Click the **Testing** tab.

WebLogic Server Administration Console displays an URL for the BRM web services.

Deploying and Launching Web Services Manager on Tomcat Server

You can deploy Web Services Manager on Tomcat Server through the Tomcat Web Application Manager. Depending on the type of payload supported by web services, deploy one of the following files:

- **infranetwebsvc.war**: Includes the web services that support the payload as an XML string data type. See "[Deploying and Launching infranetwebsvc.war](#)".
- **BrmWebServices.war**: Includes the web services that support the payload as an XML element data type. See "[Deploying and Launching BrmWebServices.war](#)".

If you customize web services, regenerate **infranetwebsvc.war** or **BrmWebServices.war** and use the generated version. Otherwise, you should use the default **infranetwebsvc.war** or **BrmWebServices.war** file. For more information about customizing web services, see "[Customizing Web Services](#)".

Deploying and Launching infranetwebsvc.war

To deploy Web Services Manager for web services that support the payload as an XML string data type on Tomcat server:

1. Create the Tomcat server domain.
See the Tomcat documentation for detailed instructions.
2. Download the JAX-WS Reference Implementation (RI) library from the JAX-WS page (<https://javaee.github.io/metro-jax-ws/>).
3. Extract the **jaxws-ri-2.2.10.zip** file and copy the following files to *Tomcat_home/lib*, where *Tomcat_home* is the directory in which the Tomcat server is installed:
 - **gmbal-api-only.jar**
 - **jaxb-api.jar**
 - **jaxb-impl.jar**
 - **jaxws-rt.jar**
 - **management-api.jar**
 - **policy.jar**
 - **stax-ex.jar**
 - **jaxb-core.jar**
 - **ha-api.jar**
 - **jaxws-tools.jar**
 - **jaxb-xjc.jar**
 - **jaxb-impl.jar**
 - **jaxb-jxc.jar**
 - **streambuffer.jar**
4. In the **War file to deploy** section, click **Browse**.
5. Click **Deploy**.
Tomcat Web Application Manager displays the deployed application in the **Applications** list.

Deploying and Launching BrmWebServices.war

To deploy Web Services Manager for web services that support the payload as an XML element data type on Tomcat server:

1. Create the Tomcat server domain.
See the Tomcat documentation for detailed instructions.
2. Download the JAX-WS RI library from the JAX-WS Reference Implementation page (<https://javaee.github.io/metro-jax-ws/>).
3. Extract the **jaxws-ri-2.2.10.zip** file and copy the following files to *Tomcat_home/lib*, where *Tomcat_home* is the directory in which Tomcat server is installed:
 - **gmbal-api-only.jar**

- **jaxb-api.jar**
 - **jaxb-impl.jar**
 - **jaxws-rt.jar**
 - **management-api.jar**
 - **policy.jar**
 - **stax-ex.jar**
 - **jaxb-core.jar**
 - **ha-api.jar**
 - **jaxws-tools.jar**
 - **jaxb-xjc.jar**
 - **jaxb-impl.jar**
 - **jaxb-jxc.jar**
 - **streambuffer.jar**
4. Download **xalan-2.7.0.jar** from the Download xalan-2.7.0.jar web page (<http://www.java2s.com/Code/Jar/x/Downloadxalan270jar.htm>) and copy the file to *Tomcat_home/lib*.
 5. If you plan to deploy the **BrmWebServices.war** file, do the following:
 - a. Open the **BrmWebServices.war/WEB-INF/web.xml** in a text editor.
 - b. Uncomment the serverlet-to-URL mapping.
 - c. Save and close the file.
 6. Copy the *BRM_home/deploy/web_services/BrmWebServices.war* file to a local directory on the machine on which you installed Tomcat server, where *BRM_home* is the directory in which BRM is installed.
 7. Log in to the Tomcat Web Application Manager.
 8. In the **War file to deploy** section, click **Browse**.
 9. Select the **BrmWebServices.war** file.
 10. Click **Deploy**.

Tomcat Web Application Manager displays the deployed application in the **Applications** list.

To launch Web Services Manager for web services that support the payload as an XML element data type, from the Tomcat Web Application Manager:

1. In the Applications list, click the **/BrmWebServices** link.
2. The Tomcat Web Application Manager displays an HTTP and an HTTPS URL for the BRM web services.

A sample URL for the **BRMCUSTServices_v2** Web service is as follows:

```
http://ipaddress:port/BrmWebServices/BRMCUSTServices_v2?wsdl
```

where:

- *ipaddress* is the domain IP address of the application server on which Web Services Manager is deployed.

- *port* is the domain port number of the application server on which Web Services Manager is deployed.

Web Services Manager displays the WSDL URLs for each available service.

For more information about the BRM web services included in Web Services Manager that take the payload as an XML element data type, see "[Customizing Web Services](#)".

4

Configuring Web Services Manager

Learn how to configure Oracle Communications Billing and Revenue Management (BRM) Web Services Manager by connecting the deployed application to the BRM system and configuring security, authorization, and Java logging for the deployed application.

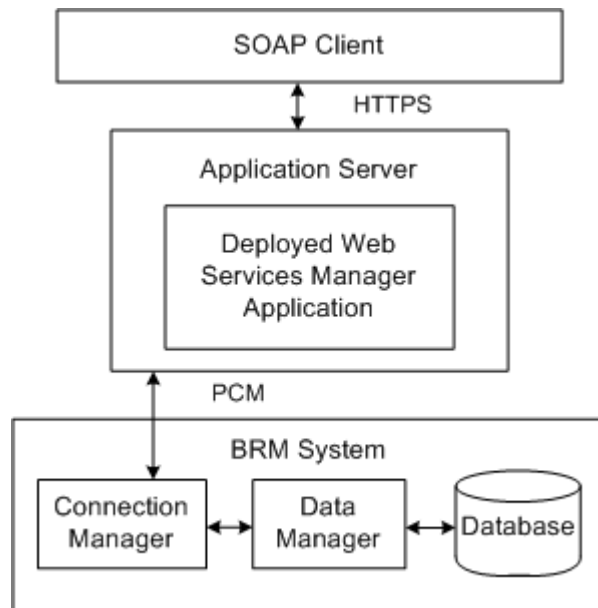
Topics in this document:

- [About Connecting Web Services Manager to the BRM System](#)
- [Configuring Security for Web Services Manager](#)
- [Disabling the JarScanner Feature in Tomcat Server](#)
- [Configuring Java Logging for the Application Server](#)

About Connecting Web Services Manager to the BRM System

Web Services Manager connects to the BRM system through a BRM Connection Manager (CM). [Figure 4-1](#) shows how BRM and the SOAP client communicate with the deployed application. Web Services Manager translates Portal Communication Module (PCM) communications sent from a CM in the BRM system into SOAP requests sent to the SOAP client over HTTPS. Web Services Manager translates SOAP responses sent from the SOAP client over HTTP into PCM communications that are returned to the CM.

Figure 4-1 Architecture of Web Services Manager in the BRM System



Connecting Web Services Manager to the BRM System

Before you connect Web Services Manager to the BRM system, verify that your instance of Web Services Manager is deployed to an application server.

If you customized Web services, use the custom **infranetwebsvc.war** or **BrmWebServices.war** file. Otherwise, you should use the default **infranetwebsvc.war** or **BrmWebServices.war** file. For more information about customizing Web services, see "[Customizing Web Services](#)".

To connect Web Services Manager to the BRM system:

1. On your application server, copy the `BRM_home/deploy/web_services/Infranet.properties` file to one of the following:
 - `local_dir/WEB-INF/classes` directory, where `local_dir` is a directory on the machine on which you installed your application server.

Note:

If you copy the **Infranet.properties** file to the `local_dir/WEB-INF/classes` directory, extract the **infranetwebsvc.war** file or **BrmWebServices.war** file to a local directory (`local_dir`) on the system on which your application server is installed.

- The home directory on the machine on which you installed your application server.
2. Open the **Infranet.properties** file in a text editor.
 3. If your BRM server and Web Services Manager instances are running on the same server, update these parameters:

```
infranet.connection=pcp://root.0.0.0.1:password@ipAddress:port/0.0.0.1/service/
admin_client 1
infranet.login.type=1
```

where:

- `password` is the password for the BRM server.
 - `ipAddress` is the IP address of the system on which BRM is installed.
 - `port` is the port number used by the application server on which BRM is installed.
4. If your BRM server is running on a different server than Web Services Manager, comment out the **infranet.connection** parameter and **add** the **infranet.wallet.location** parameter:

```
#infranet.connection=
infranet.wallet.location=wallet_location
```

where `wallet_location` is the PCP connection to your BRM server with the path to your client Oracle wallet. For example: **pcp://root.0.0.0.1:password@ipAddress:port/0.0.0.1/service/admin_client/scratch/pin00/WALLET.**

5. If SSL is enabled in the CM, update these parameters:

```
infranet.pcp.ssl.enabled=true
infranet.pcp.ssl.wallet.location=wallet_directory
```


where *wallet_directory* is the path to your client Oracle wallet. The client Oracle wallet contains the optional client SSL certificate and the private key, and it contains the Trusted CA certificate.

6. If you added custom opcodes or custom fields for Web services, add the enum values of the custom fields.

For example, if you created the **custom_fld_usage_id** field and the enum value for the **custom_fld_usage_id** field is 10001, add this entry:

```
infranet.custom.field.10001=custom_fld_usage_id
```

For information about mapping enum values, see "Creating Custom Fields" in *BRM Developer's Guide*.

7. (Optional) To configure the connection pool parameters, modify the following entries:

```
infranet.connectionpool.minsize=min_connections
infranet.connectionpool.maxsize=max_connections
infranet.connectionpool.timeout=connection_timeout
infranet.connectionpool.maxrequestlistsize=connection_maxrequest
infranet.connectionpool.maxidletime=connection_maxidle
```

where:

- *min_connections* is the minimum number of connections allowed in the pool. The default number is **1**.
 - *max_connections* is the maximum number of connections allowed in the pool. The default number is **8**.
 - *connection_timeout* is the connection pool timeout in milliseconds. The default value is **30000 milliseconds**.
 - *connection_maxrequest* is the maximum number of connection requests the connection pool can queue before returning an error. The default number is **50**.
 - *connection_maxidle* is the time in milliseconds that an idle (unused) connection remains in the connection pool before it is removed. The default value is **10000 milliseconds**.
8. (Optional) To configure logging for Web Services Manager, modify the following entry:

```
webservices.log.enabled=log_value
```

where *log_value* is one of the following:

- **true** enables logging. This option saves and displays the log files as standard output in the application server console.
 - **false** disables logging. This option saves the log files in the **/domain/logs/BRMWebSvcMgr.log** file. Configure the **BRM_home/deploy/web_services/lib/weblogic_ws_startup.jar** file to use this option.
9. Save and close the file.
 10. If you are working in the *local_dir/WEB-INF/classes* directory, regenerate the WAR file by running one of the following commands:

- To regenerate the **infranetwebsvc.war** file:

```
jar -cvf infranetwebsvc.war *
```

- To regenerate the **BrmWebServices.war** file:

```
jar -cvf BrmWebServices.war *
```

11. Deploy the regenerated **infranetwebsvc.war** or **BrmWebServices.war** file on the server. See "Deploying Web Services Manager".

Changing the Instance of BRM to which Web Services Manager Connects

If you customized Web services, use the custom **infranetwebsvc.war** or **BrmWebServices.war** file. Otherwise, you should use the default **infranetwebsvc.war** or **BrmWebServices.war** file. For more information about customizing Web services, see "Customizing Web Services".

To change the instance of BRM to which Web Services Manager connects:

1. On your application server, copy the *BRM_home/***deploy/web_services/Infranet.properties** file to one of the following:
 - *local_dir/WEB-INF/classes* directory, where *local_dir* is a directory on the machine on which you installed your application server.

Note:

If you copy the **Infranet.properties** file to the *local_dir/WEB-INF/classes* directory, extract the **infranetwebsvc.war** or **BrmWebServices.war** file to a local directory (*local_dir*) on the system on which your application server is installed.

- The home directory on the machine on which you installed your application server.
2. Open the copied **Infranet.properties** file.
 3. If your BRM server and Web Services Manager instances are running on the same server, update these parameters:

```
infranet.connection=pcp://root.0.0.0.1:password@ipAddress:port/0.0.0.1/service/  
admin_client 1  
infranet.login.type=1
```

where:

- *password* is the password for the BRM server.
 - *ipAddress* is the IP address of the system on which BRM is installed.
 - *port* is the port number used by the application server on which BRM is installed.
4. If your BRM server is running on different server than Web Services Manager, comment out the **infranet.connection** parameter and **add** the **infranet.wallet.location** parameter:

```
#infranet.connection=  
infranet.wallet.location=wallet_location
```

where *wallet_location* is the PCP connection to your BRM server with the path to your client Oracle wallet. For example: **pcp://root.0.0.0.1:password@ipAddress:port/0.0.0.1/service/admin_client/scratch/pin00/WALLET**.

5. If SSL is enabled in the Connection Manager (CM), locate the following lines and update the parameters if necessary:

```
infranet.pcp.ssl.enabled=true  
infranet.pcp.ssl.wallet.location=wallet_directory
```

where *wallet_directory* is the path to your client Oracle wallet. The client Oracle wallet contains the optional client SSL certificate and the private key, and it contains the Trusted CA certificate.

6. If you added custom opcodes or custom fields for Web services, add the enum values of the custom fields.

For example, if you created the **custom_fld_usage_id** custom field and the enum value for the **custom_fld_usage_id** field is 10001, add the following entry:

```
infranet.custom.field.10001=custom_fld_usage_id
```

For information about mapping enum values, see "Creating Custom Fields" in *BRM Developer's Guide*.

7. (Optional) To configure the connection pool parameters, modify the following entries:

```
infranet.connectionpool.minsize=min_connections
infranet.connectionpool.maxsize=max_connections
infranet.connectionpool.timeout=connection_timeout
```

where:

- *min_connections* is the minimum number of connections allowed in the pool. The default number is **1**.
- *max_connections* is the maximum number of connections allowed in the pool.
- *connection_timeout* is the connection pool timeout in milliseconds.

8. (Optional) To configure logging for Web Services Manager, modify the following entry:

```
webservices.log.enabled=log_value
```

where *log_value* is one of the following:

- **true** enables logging. This option saves and displays the log files as standard output in the application server console.
- **false** disables logging. This option saves the log files in the **/domain/logs/BRMWebSvcMgr.log** file. Configure the **BRM_home/deploy/web_services/lib/weblogic_ws_startup.jar** file to use this option.

9. (Optional) To configure searching in BRM using the PCM_OP_SEARCH opcode, restrict the PCM_OP_SEARCH opcode to pre-defined search templates by modifying the following entry:

```
allowed.search.template.ids=template_id
```

where *template_id* is the template ID of the search template that you want the PCM_OP_SEARCH opcode to use for searching. Use a comma (,) to separate multiple template IDs. If you do not want to restrict the PCM_OP_SEARCH opcode to any pre-defined search templates, set *template_id* to **None**.

For a list of template IDs, connect to the BRM database and check the list of POIDS and the respective templates in the SEARCH_T table in the BRM database. For more information, see "Searching for Objects in the BRM Database" in *BRM Developer's Guide*.

10. If you added custom opcodes or custom fields for Web services, add the enum values of the custom fields. For information about mapping enum values, see "Creating Custom Fields" in *BRM Developer's Guide*.

For example, if you created the **custom_fld_usage_id** field and the enum value for the **custom_fld_usage_id** field is 10001, add the following entry:

```
infranet.custom.field.10001=custom_fld_usage_id
```

11. Save and close the file.
12. If you are working in the *local_dir*/WEB-INF/classes directory, regenerate the WAR file by running one of the following commands:
 - To regenerate the **infranetwebsvc.war** file:

```
jar -cvf infranetwebsvc.war *
```
 - To regenerate the **BrmWebServices.war** file:

```
jar -cvf BrmWebServices.war *
```
13. Deploy the regenerated **infranetwebsvc.war** or **BrmWebServices.war** file on the server. See "[Deploying Web Services Manager](#)".

Configuring Security for Web Services Manager

By default, secure sockets layer (SSL) security for Web Services Manager is enabled. If you disabled SSL during the BRM server installation, you can enable SSL in Web Services Manager by configuring security parameters and enabling the SSL security feature in the application server on which Web Services Manager is deployed.

Configuring Security for Web Services Manager in WebLogic Server

Before you configure security for Web Services Manager, ensure that WebLogic Server and Web Services Manager are installed and that Web Services Manager has been deployed on a WebLogic Server domain. See "[Installing Web Services Manager](#)" and "[Deploying Web Services Manager](#)" for more information.

To configure security for Web Services Manager in WebLogic Server, do the following:

1. Configure authentication for Web Services Manager. See "[Configuring Authentication for WebLogic Server](#)".
2. Configure authorization for Web Services Manager. See "[Configuring WebLogic Security Policy on BRM Web Services for JAX-WS in WebLogic Server](#)".

Configuring Authentication for WebLogic Server

Before you configure authentication for Web Services Manager, create a user, group, and security realm for Web Services Manager in WebLogic Server. For more information about creating users and groups, see the discussion about users, groups, and security roles in *Fusion Middleware Securing Resources Using Roles and Policies for Oracle WebLogic Server*. For more information about security realms, see the discussion about security realms in WebLogic Server in *Fusion Middleware Securing Oracle WebLogic Server*.

To configure authentication for Web Services Manager in WebLogic Server:

1. Open the *local_dir*/infranetwebsvc.war/WEB-INF/weblogic.xml file in a text editor, where *local_dir* is a directory on the WebLogic host where you copied the **infranetwebsvc.war** file.
2. Uncomment the following lines:

```
# <security-role-assignment>
#   <role-name>brmws</role-name>
#   <externally-defined/>
# </security-role-assignment>
```
3. Save and close the file.

4. Open the `local_dir\infranetwebsvc.war\WEB-INF\web.xml` file in a text editor.
5. Uncomment the following lines:

```
# <security-constraint>
# <web-resource-collection>
# <web-resource-name>restricted web services</web-resource-name>
# <url-pattern>*/</url-pattern>
# <http-method>GET</http-method>
# <http-method>POST</http-method>
# </web-resource-collection>
# <auth-constraint>
# <role-name>brmws</role-name>
# </auth-constraint>
# <user-data-constraint>
# <transport-guarantee>CONFIDENTIAL</transport-guarantee>
# </user-data-constraint>
# </security-constraint>

# <login-config>
# <auth-method>BASIC</auth-method>
# <realm-name>default</realm-name>
# </login-config>
# <security-role>
# <role-name>brmws</role-name>
# </security-role>
```

6. Save and close the file.
7. Log in to WebLogic Server Administration Console.
8. Expand **Environment** and then select **Servers**.
The Summary of Servers page appears.
9. Select the server for which you want to enable the SSL port.
10. Click the **Configuration** tab.
11. Click the **General** subtab.
12. Select the **SSL Listen Port Enabled** check box.
13. In the **SSL Listen Port** field, enter a free port number. The default is **7002**.
14. Click **Save**, which configures Web Services Manager with the following default port numbers:

- HTTP connection: 7001

Web services that take the payload as an XML element have the default connection set to HTTP.

- HTTPS connection: 7002

If you use a SOAP development application to generate a Web service client and use port numbers other than the default port numbers, the URLs for the Web services that take the payload as an XML element will show port numbers that do not match the port numbers you configured in WebLogic Server Administration Console. Populate the correct port numbers in the URLs for the WSDL files that are generated dynamically by your SOAP development application by changing the port numbers manually in your SOAP development application request.

Configuring WebLogic Security Policy on BRM Web Services for JAX-WS in WebLogic Server

You define access restrictions for Web services in security policies in WebLogic Server.

To configure WebLogic Security Policy on BRM Web Services for JAX-WS in WebLogic Server:

1. Log in to WebLogic Server Administration Console.
2. In the Domain Structure pane, click **Deployments** and click the **BrmWebServices** link. A list of all the web services is displayed.
3. Click any of the web service.
4. Click the **Configuration** tab.
5. Click the **WS-Policy** subtab.
6. Click the WS-Policy files associated with this web service.
7. Select **WebLogic** in the **Configure the Policy Type for a Web Service** section.
8. Add the policies from the **Available Endpoint Policies** for the selected service.

If you want to use the policy for HTTPS with basic authentication, add **policy:policy_name-Https-BasicAuth.xml**, where *policy_name* is name of the policy for the selected service; for example, policy:Wssp1.2-2007-Https-BasicAuth.xml.

If you want to use the policy for HTTPS without authentication, add **policy:policy_name-Https.xml**, where *policy_name* is name of the policy for the selected service; for example, policy:Wssp1.2-2007-Https.xml.

9. Click **Finish**.
10. Click **OK** in the **Save Deployment Plan Assistant** section.
If you have multiple deployments, then the **plan.xml**, which is created when you assign a policy to the service, should be saved in its respective deployment directory.
11. Click the **Security** tab.
12. Click the **Policies** subtab.
13. In the **Web Service Methods** list, select the web method that you want to secure.
14. Click **Add Conditions**.
15. In the **Predicate List** list, select one of the following: Roles, Users, or Groups.
16. Click **Next**.
17. In the **User Argument Name** field, add the user/group.
18. Click **Add**.
19. Click **Finish**.
20. Click **Save**.

If you have enabled SSL, add the following entry in the **BRMWebServices.war/WEB-INF/web.xml** file to enable cookie security:

```
<cookie-config>
    <secure>true</secure>
</cookie-config>
```

**Note:**

This entry should be added in the session-config element of the **BRMWebServices.war/WEB-INF/web.xml** file.

Configuring Security for Web Services Manager in Tomcat Server

Before you configure security for Web Services Manager, ensure that Tomcat server and Web Services Manager are installed and that Web Services Manager has been deployed on a Tomcat server domain. See "[Installing Web Services Manager](#)" and "[Deploying Web Services Manager](#)" for more information.

To configure security for Web Services Manager in Tomcat server, do the following:

1. Configure authentication for Web Services Manager for JAX-WS in Tomcat server. See "[Configuring Authentication for Web Services Manager for JAX-WS in Tomcat Server](#)".
2. Enable SSL in Tomcat server. See "[Enabling SSL in Tomcat Server](#)".

Configuring Authentication for Web Services Manager for JAX-WS in Tomcat Server

To configure authentication for Web Services Manager for JAX-WS in Tomcat server:

1. Open the *local_dir*/**BrmWebServices.war/WEB-INF/web.xml** file in a text editor.
2. Add the following lines:

```
# <security-constraint>
# <web-resource-collection>
# <web-resource-name>restricted web services</web-resource-name>
# <url-pattern>/*</url-pattern>
# <http-method>GET</http-method>
# <http-method>/POST</http-method>
# </web-resource-collection>
# <auth-constraint>
# <role-name>brmws</role-name>
# </auth-constraint>
# <user-data-constraint>
# <transport-guarantee>CONFIDENTIAL</transport-guarantee>
# </user-data-constraint>
# </security-constraint>

# <login-config>
# <auth-method>BASIC</auth-method>
# </login-config>
# <security-role>
# <role-name>brmws</role-name>
# </security-role>
```

3. Save and close the file.
4. Open the *local_dir*/**apache-tomcat-version/conf/tomcat-users.xml** file in a text editor.
5. Locate the following lines and specify the login details of the user:

```
<role rolename="brmws"/>
<user username="username" password="password" roles="brmws"/>
```

where:

- *username* is the username for accessing Web services.
 - *password* is the password for accessing Web services.
6. Save and close the file.
 7. Open the **config/server.xml** file in a text editor.
 8. In the **<Engine>** section, add the following class path:

```
<Realm className="org.apache.catalina.realm.MemoryRealm" />
```

9. Save and close the file.
10. Restart the Tomcat server.

Enabling SSL in Tomcat Server

To enable secure communication for Web Services Manager, enable secure sockets layer (SSL) in the Tomcat server domain on which you deploy Web Services Manager.

To enable SSL for Tomcat server:

1. Generate the keystore by running the following command:

```
keytool -genkey -alias mykes -keyalg RSA -keystore mykeystore
```

where:

- *mykes* is the alias.
- *mykeystore* is the name of the keystore.

2. Open the **conf/server.xml** file in a text editor.
3. Uncomment the following lines and specify the path for the keystore file:

```
# <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
# address="IPAddress"
# maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
# clientAuth="false" sslProtocol="TLS"
# keystoreFile="filepath"
# keystorePass="password" />
```

where:

- *IPAddress* is the IP address of the machine on which you installed the Apache Tomcat server.
- *filepath* is the keystore file path.
- *password* is the password for the keystore file.

4. Save and close the file.

Disabling the JarScanner Feature in Tomcat Server

The JarScanner feature in the Tomcat server is used to scan the web application for JAR files. To avoid unnecessary warnings displayed for optional JAR files, disable the JarScanner feature in the Tomcat server.

To disable the JarScanner feature in the Tomcat server:

1. Open the *local_dir/apache-tomcat-version/conf/context.xml* in a text editor.
2. Search for the following entry:


```
<JarScanner scanClassPath="true" scanAllFiles="false" scanAllDirectories="false"></JarScanner>
```

3. Set the scanClassPath entry to **false**:

```
<JarScanner scanClassPath="false" scanAllFiles="false" scanAllDirectories="false"></JarScanner>
```

4. Save and close the file.

Configuring Java Logging for the Application Server

Depending on your configuration, you may wish to change the level of Java logging on the application server. To configure the Java logging level, do the following:

- For WebLogic Server, see "[Configuring Java Logging for WebLogic Server](#)" for Web Services Manager-specific configuration. For more information, see the discussion about application logging and WebLogic logging services in *Fusion Middleware Using Logging Services for Application Logging for Oracle WebLogic Server*.
- For Tomcat server, see the discussion about logging in Tomcat in *Tomcat User Guide*.

Configuring Java Logging for WebLogic Server

To configure Java logging in WebLogic Server:

1. Specify the Java Unified Logging (JUL) mechanism. See "[Specifying the Java Unified Logging \(JUL\) Mechanism](#)".
2. Create a startup class. See "[Creating a Startup Class](#)".

Specifying the Java Unified Logging (JUL) Mechanism

Specifying the JUL mechanism allows Web Services Manager to use JUL in addition to the WebLogic Server Administration Console logging.

To specify the JUL mechanism:

1. Open the `BRM_home/deploy/web_services/Infranet.properties` file in a text editor.
2. Uncomment the following entry:

```
# webservices.log.enabled = true
```

3. Change the value to **false**:

```
webservices.log.enabled = false
```

4. Save and close the file.

Creating a Startup Class

You define a startup class to enable JUL and create log files for the following Web service classes:

- `com.portal.webservices.BRMFlistToXML`
- `com.portal.webservices.BRMXMLToFlist`
- `com.portal.webservices.OpcodeCaller`
- `com.portal.webservices.WebServicesUtilities`

To create a startup class:

1. Copy the *BRM_home/deploy/web_services/weblogic_ws_startup.jar* file to the *server_domain/lib* directory, where *server_domain* is the WebLogic Server domain in which Web Services Manager is deployed.
2. Log in to WebLogic Server Administration Console.
3. Click **Lock and Edit**.
4. In the **Domain Structure** pane, expand **Environment** and then click **Startup and Shutdown classes**.
The Startup and Shutdown Classes pane appears.
5. Click **New**.
The Configure a New Startup or Shutdown Class: Class Type pane appears.
6. Select **Startup Class** and click **Next**.
The Configure a New Startup or Shutdown Class: Startup Class Properties pane appears.
7. In the **Name** field, enter **BRMWSLoggerStartUpClass**.
8. In the **Class Name** field, enter **com.portal.webservices.BRMWSLoggerStartUp**.
9. In the **Argument** field, set the log level. This field sets the log level for all the classes in Web Services Manager:
 - To log problems that require attention from the system administrator, enter **SEVERE**. This is the default.
 - To log the most detailed trace and debug messages, enter **FINEST**.
 - To log highly detailed trace and debug messages, enter **FINER**.
 - To log trace and debug messages for performance monitoring, enter **FINE**.
10. Click **Next**.
The Configure a New Startup or Shutdown Class: Select Targets pane appears.
11. From the **Servers** list, select the server on which to deploy the class.
The Startup and Shutdown Classes pane appears.
12. Click **Finish**.
13. Click **BRMWSLoggerStartUpClass**.
The Settings for BRMWSLoggerStartUpClass pane appears.
14. Select **Run Before Application Deployments** and **Run Before Application Activations** and click **Save**.
15. Click **Activate Changes**.
16. Restart the WebLogic Server, which applies changes.
17. Redeploy any existing Web Services Manager deployments. See "[Deploying Web Services Manager](#)".

By default, log files are created in the *WebLogic_home/user_projects/domains/domain_name/logs/BRMWebServicesMgrLogs/BRMWebServicesMgr.log* file

where:

- *WebLogic_home* is the directory in which WebLogic Server is installed.
- *domain_Name* is the name of the domain you are configuring.

5

Securing Web Services Manager with OAuth 2.0

Learn how to secure Oracle Communications Billing and Revenue Management (BRM) Web Services Manager with the OAuth 2.0 authorization framework.

Topics in this document:

- [About the OAuth 2.0 Authorization Framework](#)
- [Setting Up Web Services Manager with OAuth 2.0](#)
- [Sending SOAP Requests to BRM Web Services](#)

About the OAuth 2.0 Authorization Framework

Web Services Manager uses the OAuth 2.0 protocol to authenticate a client application's identity and to authorize the client application to access BRM Web services. It does this by validating an OAuth access token that is passed in the header of the client's HTTP/HTTPS request to Web Services Manager.

Your client must pass this OAuth access token in the header of every HTTP/HTTPS request sent to Web Services Manager.

Setting Up Web Services Manager with OAuth 2.0

To set up your client application to use OAuth 2.0 authentication to access BRM Web services:

1. Install the Oracle Access Management software. For the list of supported versions, see "Additional BRM Software Requirements" in *BRM Compatibility Matrix*.
For information about installing the Oracle Access Management software, see [Oracle Fusion Middleware Installing and Configuring Oracle Identity and Access Management](#).
2. Create an identity domain in Oracle Access Management. See "[Creating an OAuth Identity Domain](#)".
3. Create a resource server in your identity domain. See "[Creating a Resource Server](#)".
4. Create an OAuth client in your identity domain. See "[Creating an OAuth Client](#)".
5. Validate that OAuth 2.0 is set up properly in Web Services Manager. See "[Validating Your OAuth Setup](#)".
6. Configure Web Services Manager to protect BRM Web services through Oracle Access Management. See "[Configuring Web Services Manager](#)".
7. Enable OAuth 2.0 validation in Web Services Manager. See "[Enabling OAuth Validation for Web Services Manager](#)".

Creating an OAuth Identity Domain

You create an OAuth identity domain to control the authentication and authorization of users who can sign in to Web Services Manager, and what features they can access. You create all artifacts, such as the resource server and OAuth client, under the identity domain.

To create an identity domain, use cURL to send an HTTP/HTTPS request to the Oracle Access Management URL:

```
curl -i -H "Content-Type: application/json" \  
-H "Accept: application/json" \  
-H "Authorization:Basic encoded_admin" \  
-X POST http://oam_adminHost:oam_adminPort/oam/services/rest/ssa/api/v1/  
oauthpolicyadmin/oauthidentitydomain \  
-d '{"name": "identity_domain", "description": "Description", "tokenSettings":  
[ { "tokenType": "ACCESS_TOKEN", "tokenExpiry": 3600 } ] }'
```

where:

- *encoded_admin* is the Base64-encoded format of the client ID and client secret separated by a colon (*client_id:client_secret*).
- *oam_adminHost:oam_adminPort* is the host name and port for the Oracle Access Management administration server.
- *identity_domain* is the name of the Oracle Access Management identity domain that you want to create.

For more information about the Oracle Access Management endpoint, see ["Add a new OAuth Identity Domain"](#) in *REST API for OAuth in Oracle Access Manager*.

Creating a Resource Server

A resource server hosts the protected resources. It must be capable of accepting and responding to resource requests using OAuth access tokens.

To create a resource server, use cURL to send an HTTP/HTTPS request to the Oracle Access Management URL:

```
curl -i -H "Authorization:Basic encoded_admin" \  
-H "Content-Type: application/json" \  
-H "Accept: application/json" \  
-X POST http://oam_adminHost:oam_adminPort/oam/services/rest/ssa/api/v1/  
oauthpolicyadmin/application \  
-d '{ "name": "resource_server", "idDomain": "identity_domain",  
"description": "Description", "scopes": [ { "scopeName": "OAUTH1",  
"description": "All Access" } ] }'
```

where *resource_server* is the name of the resource server that you want to create.

For more information about the Oracle Access Management endpoint, see ["Add a new Resource Server"](#) in *REST API for OAuth in Oracle Access Manager*.

Creating an OAuth Client

To create an OAuth client, use cURL to send an HTTP/HTTPS request to the Oracle Access Management URL:

```
curl -i -H "Authorization:Basic encoded_admin" \  
-H "Content-Type: application/json" \  
-H "Accept: application/json" \  
-X POST http://oam_adminHost:oam_adminPort/oam/services/rest/ssa/api/v1/  
oauthpolicyadmin/client \  
-d '{ "secret": "client_secret", "id": "client_id", "name": "client_name",  
"scopes": [ "BrmWebServices.OAUTH1" ], \  
"clientType": "CONFIDENTIAL_CLIENT", "idDomain": "identity_domain",  
"description": "Description", "grantTypes":[ "CLIENT_CREDENTIALS" ],  
"defaultScope": "BrmWebServices.OAUTH1", \  
"redirectURIs": [ { "url":"http://wsm_host:wsm_port/BrmWebServices",  
"isHttps": false } ] }'
```

where:

- *client_id* and *client_secret* are the client ID and client secret.
- *client_name* is the name of the OAuth client that you want to create.
- *wsm_host:wsm_port* is the hostname and port number of the Web Services Manager server.

For more information about the Oracle Access Management endpoint, see ["Add a new OAuth Client"](#) in *REST API for OAuth in Oracle Access Manager*.

Validating Your OAuth Setup

To validate that Web Services Manager has been successfully secured with OAuth 2.0:

1. Generate an OAuth access token by submitting a POST request to the **Create Access Token Flow** endpoint in the Oracle Access Management OAuth REST API using cURL:

```
curl -i -H 'Authorization: Basic encoded_admin' \  
-H "Content-Type: application/x-www-form-urlencoded;charset=UTF-8" \  
-H "X-OAUTH-IDENTITY-DOMAIN-NAME: identity_domain" \  
--request POST http://oam_managedServerHost:oam_managedServerPort/oauth2/  
rest/token \  
-d 'grant_type=CLIENT_CREDENTIALS&scope=BrmWebServices.OAUTH1'
```

where *oam_managedServerHost* and *oam_managedServerPort* port is the host name and port for the Oracle Access Management server.

If successful, the response code 200 is returned with the access token and its expiration time in the response payload.

For more information, see ["Create Access Token Flow"](#) in *REST API for OAuth in Oracle Access Manager*.

2. Validate the access token by submitting a GET request to the **Validate Access Token Flow** endpoint in the Oracle Access Management OAuth REST API using cURL:

```
curl -i -H "X-OAUTH-IDENTITY-DOMAIN-NAME: identity_domain" \
--request GET "http://oam_managedServerHost:oam_managedServerPort/oauth2/
rest/token/info?access_token=access_token"
```

where `access_token` is the access token returned in step 1.

If successful, the response code 200 is returned with details about the access token in the response payload.

For more information, see "[Validate Access Token Flow](#)" in *REST API for OAuth in Oracle Access Manager*.

Configuring Web Services Manager

Configure Web Services Manager to protect BRM Web services through Oracle Access Management.

To configure Web Services Manager:

1. Copy the `BRM_home/deploy/web_services/Infranet.properties` file to the directory in which Web Services Manager is installed.
2. Open the copied `Infranet.properties` file in a text editor.
3. Edit the following parameters: `infranet.OAuthAccessTokenUrl:http://oam_host:oam_port/oauth2/rest/ token/info` `infranet.OAuthDomainName:identity_domain`.

```
infranet.OAuthOldOAM=false
infranet.OAuthAccessTokenUrl:http://oam_host:oam_port/oauth2/rest/ token/
info
infranet.OAuthDomainName:identity_domain
```

4. Save and close the file.
5. Restart your WebLogic Server.

When you restart the WebLogic Server, ensure that the `libportal.so` BRM library is set in `LD_LIBRARY_PATH`. For JRE on 64-bit environments, rename `libportal64.so` to `libportal.so`.

Enabling OAuth Validation for Web Services Manager

To enable OAuth validation for BRM web services:

1. Go to the `BRM_home/deploy/web_services` directory and then extract the contents of the `BrmWebServices.war` file to a local directory (`local_dir`):

```
jar -xvf BrmWebServices.war
```

2. Open the `local_dir/WEB-INF/web.xml` file in a text editor.
3. Uncomment these `filter` and `filter-mapping` tags:

```
<filter>
  <filter-name>OAuthTokenValidationFilter</filter-name>
  <filter-class>com.portal.jax.OAuthTokenValidationFilter</filter-
```

```

class>
</filter>
<filter-mapping>
  <filter-name>OAuthTokenValidationFilter</filter-name>
  <servlet-name>BrmWebServices</servlet-name>
  <url-pattern>/BrmWebServices/*</url-pattern>
  <url-pattern>/BRMPricingServices_v2</url-pattern>
  <url-pattern>/BRMBalServices_v2</url-pattern>
  <url-pattern>/BRMARServices_v2</url-pattern>
  <url-pattern>/BRMBillServices_v2</url-pattern>
  <url-pattern>/BRMCustServices_v2</url-pattern>
  <url-pattern>/BRMCustcareServices_v2</url-pattern>
  <url-pattern>/BRMInvServices_v2</url-pattern>
  <url-pattern>/BRMPymtServices_v2</url-pattern>
  <url-pattern>/BRMCollectionServices_v2</url-pattern>
  <url-pattern>/BRMReadServices_v2</url-pattern>
  <url-pattern>/BRMActServices_v2</url-pattern>
  <url-pattern>/BRMSubscriptionServices_v2</url-pattern>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>

```

4. Save and close the file.
5. Regenerate the **BrmWebServices.war** file.
 - a. Go to *local_dir* and delete the existing **BRMWebServices.war** file:

```

cd local_dir
rm BrmWebServices.war

```

- b. Create a new **BrmWebServices.war** archive file:

```

jar -cvf BrmWebServices.war .

```

Sending SOAP Requests to BRM Web Services

After you have set up OAuth 2.0 authentication in Web Services Manager, you can start submitting SOAP requests to the BRM Web services.

To send a request to a BRM Web service:

1. Submit a GET request to the BRM Web service that you want to use:

```

curl -i -H "X-OAUTH-IDENTITY-DOMAIN-NAME: identity_domain" \
-H "Authorization:Bearer access_token" \
--request GET http://wsm_wlsHost:wsm_wlsPort/BrmWebServices/
webServicesName?wsdl

```

where:

- *wsm_wlsHost:wsm_wlsPort* is the hostname and port number for the WebLogic Server that was deployed from BRM Web Services Manager.

- *webServicesName* is the name of the Web service such as **BRMACTServices_v2**, **BRMCUSTServices_v2**, or **BRMPYMTServices_v2**. For the Web service names, see "[About WSDL Files and BRM Opcodes](#)".
2. Submit a request to the target SOAP operation, ensuring that you send the OAuth access token in the header request:

```
curl -i -H "Content-Type: text/xml;charset=UTF-8"
--request POST http://wsm_wlsHost:wsm_wlsPort/BrmWebServices/
webServicesName
-d '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:bus="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">
<soapenv:Header>
  <code>access_token</code>
</soapenv:Header>
<soapenv:Body>
  <bus:operationNameRequest>
    <bus:opcode_Request>
      <bus:opcode_inputFlist>
        </bus:opcode_inputFlist>
      </bus:opcode_Request>
    </bus:pcmOpSearchRequest>
  </soapenv:Body>
</soapenv:Envelope>'
```

where:

- *operationName* is the name of the SOAP operation to call in the Web service interface.
- *opcode* is the name of the BRM opcode to call.

For example, to search for accounts that have purchased a particular package, you would submit this request to the **pcmOpSearch** operation in the **BRMReadServices_v2** interface:

```
curl -i -H "Content-Type: text/xml;charset=UTF-8"
--request POST http://wsm_wlsHost:wsm_wlsPort/BrmWebServices/
BRMReadServices_v2
-d '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:bus="http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes">
<soapenv:Header>
  <code>access_token</code>
</soapenv:Header>
<soapenv:Body>
  <bus:pcmOpSearchRequest>
    <bus:PCM_OP_SEARCH_Request>
      <bus:flags>0</bus:flags>
      <bus:PCM_OP_SEARCH_inputFlist>
        <bus:FLAGS>256</bus:FLAGS>
        <bus:ARGS elem="1">
          <bus:POID>0.0.0.1 /plan -1 0</bus:POID>
        </bus:ARGS>
        <bus:ARGS elem="2">
          <bus:ACCOUNT_OBJ>0.0.0.1 /account 1 0</bus:ACCOUNT_OBJ>
        </bus:ARGS>
        <bus:ARGS elem="3">
```



```
        <bus:NAME>Plan 1 - Measured Web Access with Discounts</
bus:NAME>
        </bus:ARGS>
        <bus:POID>0.0.0.1 /search/pin 45 0</bus:POID>
        <bus:RESULTS elem="0"></bus:RESULTS>
        <bus:TEMPLATE>select X from /plan where F1 like V1 and F2 = V2
and F3 =V3</bus:TEMPLATE>
        </bus:PCM_OP_SEARCH_inputFlist>
        </bus:PCM_OP_SEARCH_Request>
    </bus:pcmOpSearchRequest>
</soapenv:Body>
</soapenv:Envelope>'
```

6

Customizing Web Services

Learn how to customize Oracle Communications Billing and Revenue Management (BRM) Web Services Manager to expose your custom opcodes or support custom web services.

Topics in this document:

- [Setting Up Web Services Manager to Support Custom Opcodes](#)
- [Creating a Custom Web Service](#)
- [Generating the Schema Files for Your System](#)
- [Generating the Schema for Your Opcodes](#)

Setting Up Web Services Manager to Support Custom Opcodes

To expose custom opcodes as web services, enable Web Services Manager to support custom opcodes. For more information on custom opcodes, see "Using Custom Opcodes" in *BRM Developer's Guide*.



Note:

Before you customize an opcode for a web service in Web Services Manager, implement the custom opcode in BRM.

To enable Web Services Manager to support custom opcodes:

1. Do one of the following:

- Create the **CustomOp.java** file by entering the following command:

```
parse_custom_ops_fields -L pcmjava -I input -O output -P java_package
```

where:

- *input* is the header file you create for your custom opcodes and fields.
- *output* is the memory-mapped file or directory for the output of the script. *output* must be a directory having some correspondence with the Java package. For example, if *java_package* is in **com.portal.classFiles**, *output* must be **f:/mysource/com/portal/classFiles**.
- *java_package* is the Java package in which to put the generated classes.

For more information, see the discussion about the **parse_custom_ops_fields** utility in *BRM Developer's Guide*.

- Manually create the **CustomOp.java** file.
2. Verify that the **CustomOp.java** file contains the following:
- The opcode-name-to-opcode-number mapping for all the custom opcodes in the file.

 **Note:**

Verify that the mapping includes the full name of each opcode. If any opcode name is truncated, replace the truncated name with the full name.

- The **opToString** method, which converts opcode numbers to opcode names.
- The **stringToOp** method, which converts opcode names to opcode numbers.

The following is a sample **CustomOp.java** file:

```
public class CustomOp {
    public static final int CUSTOM_OP_ACT_INFO= 100000;
    public static final int CUSTOM_OP_READ_ACT_PRODUCT = 100001;
    public static String opToString( int op ) {          try {
        java.lang.reflect.Field[] flds =
CustomOp.class.getFields();
        for( int i = 0; i < flds.length; i++ ) {
            try {
                int val = flds[i].getInt(null);
                if( val == op ) {
                    return flds[i].getName();
                }
            } catch( IllegalAccessException e ) { continue;
            } catch( IllegalArgumentException e ) { continue; }
            } catch( SecurityException e ) {}
        return null;
    }
    public static int stringToOp( String op ) {
        try {
            java.lang.reflect.Field[] flds =
CustomOp.class.getFields();
            for( int i = 0; i < flds.length; i++ ) {
                try {
                    String name = flds[i].getName();
                    if( name.equals(op) ) {
                        return flds[i].getInt(null);
                    }
                } catch( IllegalAccessException e ) { continue;
            }
            } catch( IllegalArgumentException e ) { continue; }
            } catch( SecurityException e ) {}
        return -1;
    }
}
```

3. Compile the **CustomOp.java** file into the **CustomOp.class** file by entering the following command:

```
javac -d . path/CustomOp.java
```

For example:

```
javac -d . com/portal/classFiles/CustomOp.java
```

4. Package the **CustomOp.class** file into the **CustomFields.jar** file by entering the following command:

 **Note:**

Make sure the JRE version that was used to generate the **CustomFields.jar** file is the same or lower than the version of the WebLogic Server JRE.

```
jar -cvf CustomFields.jar path.CustomOp.class
```

For example:

```
jar cvf CustomFields.jar com.portal.classFiles.CustomOp.class
```

5. Make the **CustomFields.jar** file available to Web Services Manager by doing one of the following:
 - If you have not deployed Web Services Manager, do the following:
 - a. Copy the *path/CustomFields.jar* file to the *local_dir/WEB-INF/lib* directory, where *path* is the path to the **CustomFields.jar** file (for example, **com/portal/classFiles**).
 - b. Open the *BRM_home/deploy/web_services/Infranet.properties* file in a text editor.
 - c. Add or modify the following entry:

```
infranet.custom.field.package = package
```

where *package* is the name of the package that contains the **CustomOp.java** file; for example, **com.portal.classFiles**.
 - d. Add all the custom fields to the **Infranet.properties** file.
 - e. Save and close the file.
 - f. Copy the *BRM_home/deploy/web_services/Infranet.properties* file to the *local_dir/WEB-INF/classes* directory or the home directory on the machine on which you installed WebLogic Server.
 - If you have deployed Web Services Manager, do the following:
 - a. Copy the *path/CustomFields.jar* file to the *local_dir/WEB-INF/lib* directory.
where *local_dir* is the directory in which you deployed Web Services Manager on your application server.
 - b. Open the *Webservices_deployment_dir/WEB-INF/classes/Infranet.properties* file in a text editor.
 - c. Add or modify the following entry:

```
infranet.custom.field.package = package
```

where *package* is the name of the package that contains the **CustomOp.java** file; for example, **com.portal.classFiles**.
 - d. Add all the custom fields to the **Infranet.properties** file.
 - e. Save and close the file.

Creating a Custom Web Service

You can extend Web Services Manager to support custom web services. Before you create a custom web service or customize an existing web service in Web Services Manager, implement your custom opcodes in the BRM system. For more information, see "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.

To create a custom web service:

1. If you created an opcode with custom fields for your custom web service, configure BRM to recognize the custom fields. See "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.
2. Create a WSDL file for the web service. See "Generating WSDL Files for Web Services" in *BRM JCA Resource Adapter*.

To create a WSDL file manually, do the following:

- For web services that support payload as XML string, see the **deploy/web_services/wSDL** sample file and create the WSDL file.
 - For web services that support payload as XML element, see the **deploy/web_services/BrmWebServices.war/WEB-INF/wSDL** sample file and create the WSDL file.
3. Create the XML specifications for your custom opcodes. See "[Creating Opcode Specification Schema Files](#)".
 4. Generate web service classes for your custom service by doing the following:
 - a. Create the following directory structure in a local directory (*local_dir*) on the machine on which your application server is installed.

```
/wsdl
/src
/classes
/jar
```

- b. Copy your custom WSDL files and schema (XSD) files into the *local_dir/wsdl* directory.
- c. Copy the **BrmWebServices.war/WEB-INF/wSDL/BRMWebServiceException.xsd** file into the *local_dir/wsdl* directory.
- d. Create the **custom_services.xml** as an Ant build file.

The following is a sample **custom_services.xml** file:

```
<?xml version="1.0"?>
<project name="Custom BRM WebServices build file" default="all" basedir=".">
  <property name="buildDir" value="classes"/>
  <property name="srcDir" value="src"/>
  <property name="wsdlDir" value="wsdl"/>
  <property name="pinwsgen" value="pin_wsgen"/>

  <!-- define the classpath -->
  <path id="classpath">
    <pathelement path="${buildDir}"/>
    <pathelement path="jar/web_services.jar"/>
    <pathelement path="jar/webServicesUtils.jar"/>
  </path>
```

```

<!-- create Source files from WSDL and XSDs -->
<target name="custom_service_gen" description="Create java source files from
wsdl" >
    <exec executable="BRM_home/deploy/web-services/pin_wsgen/pin_wsgen"
failonerror="true">
        <arg value="-s"/>
        <arg value="src"/>
        <arg value="-d"/>
<arg value="${buildDir}"/>
<arg value="-p"/>
<arg value="com.portal.jax.'yourpackagesubdirname' "/>
        <arg value="${wsdlDir}/'YourCustomServices_v2.wsdl'"/>
    </exec>
</target>

<target name="all" depends="custom_service_gen, custom_jar" description="build
everything" />

<!-- compile task -->
<target name="compile" depends="custom_service_gen" description="compile source
files" >
    <echo>" Compiling JAX-WS impl classes"</echo>
    <javac srcdir="${srcDir}"
        destdir="${buildDir}"
        classpathref="classpath"
        debug="on"
        source="1.5"
    />
</target>
<!--Create custom service jar -->
<target name="custom_jar" depends="custom_service_gen, compile"
description="generate jar file" >
    <jar jarfile="custom_services.jar" basedir="${buildDir}" >
    </jar>
</target>
<!--ant clean task -->
<target name="clean" description="remove derived objects" >

<delete dir="classes/com"/>

<delete dir="custom_service.jar"/>

</target>

</project>

```

where:

- *BRM_home* is the directory in which BRM is installed.
- *YourCustomServices_v2* is the custom service WSDL file name.
- *yourpackagesubdirname* is the package directory for your custom service.

5. Generate and build your custom web services by running the following command:

```
ant -file custom_services.xml
```

6. Add all the custom field **enum** constants to the **Infranet.properties** file. See "[Connecting Web Services Manager to the BRM System](#)" for more information.
7. Package your custom web service with the **BrmWebServices.war** file by doing the following:

- a. Extract the **BrmWebServices.war** file to a local directory (*local_dir*) on the machine on which you installed your application server.
- b. Do one of the following:
 - (For WebLogic Server) Modify the *local_dir/WEB-INF/Web.xml* file to include your custom service URL mapping similar to existing URL mapping.
 - (For Apache Tomcat server) Modify the *local_dir/WEB-INF/sun.jaxws.xml* file to add your custom service implementation class.
- c. Copy your custom WSDL files and schema (XSD) files into the *local_dir/WEB-INF/wsdl/* directory.
- d. Copy your **custom_services.jar** into the *local_dir/WEB-INF/classes* directory.
- e. Copy your **CustomFields.jar** into the *local_dir/WEB-INF/lib* directory.
- f. Delete the existing **BrmWebServices.war** file.
- g. Create a new **BrmWebServices.war** file by running the following command:

```
jar -cvf BrmWebServices.war *
```

Generating the Schema Files for Your System

Web Services Manager uses schema files to validate data it sends to or receives from BRM.

To generate the schema files for your system, do the following:

1. If you modified any opcodes, generate schemas for the opcodes in your BRM system. See ["Generating the Schema for an Existing Opcode"](#).
2. Generate schemas for the storable classes and subclasses in your BRM system. See ["Generating the Schema for Your Storable Classes and Subclasses"](#) in *BRM JCA Resource Adapter*.
3. In your opcode schema files, specify the location of your storable class schema files. See ["Specifying the Location of the Storable Class Schema Files in the Opcode Schema Files"](#) in *BRM JCA Resource Adapter*.

Note:

After generating the opcode and storable class schema files, copy the schema files to a location that is accessible to the Web Services Manager. Make sure that this location is the same as the location that is specified in the **include** section of the opcode schema files and in the opcode schema **InteractionSpec** attribute in the WSDL files. See ["Specifying the Location of the Storable Class Schema Files in the Opcode Schema Files"](#) and ["Generating the WSDL Files for Your System"](#) in *BRM JCA Resource Adapter*.

Generating the Schema for Your Opcodes

The Web Services Manager package includes all the opcode schemas and list specifications you need for a default integration.

If you customized any of the opcodes that are supported by Web Services Manager or if you added support for new opcodes, you must generate XSD schema files for the opcodes.

 **Note:**

Before you customize an existing opcode specification, ensure that you update the opcode specification in the BRM system.

After you customize web services, copy the customized schema files and the WSDL files to the **infranetwebsvc.war** file.

Generating the Schema for an Existing Opcode

To generate schema files for an opcode that you customized and Web Services Manager already supports:

1. Modify the opcode's XML specification file. By default, the opcode specification XML files are installed in the *BRM_home/apps/brm_integrations/opspecs* directory, where *BRM_home* is the directory in which you installed the BRM components.
2. Do one of the following:
 - For web services that take payload as XML string:
 - Run the **pin_opspec_to_schema** utility. See "[Creating Opcode Specification Schema Files](#)".
 - Copy the customized XSD files to the *BRM_home/deploy/web_services/schemas* directory.
 - For web services that take payload as XML element:
 - Run the **pin_opspec_to_schema_v2** utility. See "[Creating Opcode Specification Schema Files](#)".
 - Copy the customized XSD files to the **infranetwebsvc/WEB-INF/services/InfranetWebservices.aar/META-INF** directory.

Creating Opcode Specification Schema Files

You must create opcode flist specification files for opcodes that you customize or add to the Web Services Manager. Create the specification XML files by following the *BRM_home/apps/brm_integrations/stylesheets/opspec.xsd* file.

You then convert the opcode flist specification XML files into XSD schema by using the **pin_opspec_to_schema** and **pin_opspec_to_schema_v2** utilities.

To convert opcode flist specification XML files into XSD schema, go to the *BRM_home/apps/brm_integrations* directory and do the following:

- For web services that take payload as XML string, run the following command:

```
pin_opspec_to_schema -i input_file [-o output_file]
```
- For web services that take payload as XML element, run the following command:

 **Note:**

Before you run the following command, specify the BRM installation directory in the **pin_opspec_to_schema_v2** utility by replacing `$PIN_HOME` with `BRM_home`.

```
pin_opspec_to_schema_v2 -i input_file > output_file
```

where:

- *input_file* specifies the name and location of the opcode's XML flist specification. By default, the utility looks for the file in the current directory.
- *output_file* creates the XSD schema output file using the name you specify. By default, the utility creates a file named *opcodename.xsd* in the directory from which you run the utility.

You can also create XSD schema for web services that take payload as XML element by using the **pin_opspec_to_schema_v2** XSD generator utility that is located in the `BRM_home/bin` directory.

To create the XSD schema file by using the **pin_opspec_to_schema_v2** utility, run the following command using Groovy:

```
groovy pin_opspec_to_schema_v2 -i input.xml > output.xsd
```

where:

- *input.xml* specifies the name of the opcode's XML flist specification
- *output.xsd* creates the XSD schema output file using the name you specify

Specifying the XSL Rules to Create the Opcode Schema

The **pin_opspec_to_schema** utility uses the `BRM_home/brm_integrations/stylesheets/pin_opspec_to_schema.xsl` style sheet to generate the schema for BRM opcodes. If your opcode references custom fields, you must customize the **pin_opspec_to_schema.xsl** style sheet to handle your custom fields.

For a list of the supported BRM data types, see "Understanding the BRM Data Types" in *BRM Developer's Guide*.