Oracle® Communications Billing and Revenue Management

Web Services Manager



Release 16.26.0
G54509-01
October 2026

ORACLE

Oracle Communications Billing and Revenue Management Web Services Manager, Release 16.26.0

G54509-01

Copyright © 2017, 2026, 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.

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.

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Contents

About This Content

1 Using Web Services

About WSDL Files and BRM Opcodes 1
Configuring BRM to Use PCM_OP_BAL_GET_ECE_BALANCES 10
Example of Reading Account Fields in BRM Using Web Services 10
Determining the WSDL URLs for Web Services Manager 10
Sample SOAP Request and Response XML Files 11
Using Metrics and Tracing 14
Working with Metrics in Web Services Manager 14
Enabling Tracing 15

2 Installing Web Services Manager

Installing Web Services Manager 1
Starting and Stopping Web Services Manager 1
Uninstalling Web Services Manager 2

3 Configuring Web Services Manager

Web Services Manager Configuration Reference 1
Validating Input and Output XML Data 5
Connecting Web Services Manager to Connection Manager 6
Restricting Stored Search Templates 7
Configuring BI Publisher for Data Export 7
Configuring Web Services Manager Logging 8
Configuring HTTPS Between SOAP Clients and Web Services Manager 8

4 Securing Web Services Manager with OAuth 2.0

About the OAuth 2.0 Authorization Framework 1
Setting Up Web Services Manager with OAuth 2.0 1
Obtaining the Token-Signing Certificate 2

Copyright © 2017, 2026, Oracle and/or its affiliates.

Configuring Web Services Manager 2
Sending SOAP Requests to BRM Web Services 2

5 Customizing Web Services Manager

Setting Up Web Services Manager to Support Custom Fields in Opcodes 1
Setting Up Web Services Manager to Support Unexposed Opcodes for XML-Element Services 1
Generating the Web Service Implementation Class for XML Element Services 3
Configuring XML-Element Service Endpoints 6
Setting Up Web Services Manager to Support Unexposed Opcodes for XML-String Services 7
Generating the Web Service Implementation Class for XML-String Services 9
Configuring XML-String Service Endpoints 11
Setting Up Web Services Manager to Support Custom Opcodes 12
Supporting Custom Opcodes for XML-Element Services 12
Supporting Custom Opcodes for XML-String Services 13
Making Custom Fields Available to Web Services Manager 13
Generating the Schema Files for Your System 15

6 Generating the Schema for Your Opcodes

Generating the Schema for an Existing Opcode 1
Creating Opcode Specification Schema Files 1
Specifying the XSL Rules to Create the Opcode Schema 2

7 Migrating to a Helidon-Based Web Services Manager

What Changes When Migrating to Helidon-Based Web Services Manager 1
Input Requests Remain Unchanged 1
Changes to Web Services Manager URLs 1
Changes to Web Services Manager Error Response Format 2
Migrating OAuth 2.0 Configuration to Helidon-Based Web Services Manager 3
Migrating Web Services Manager Customizations 3
Migrating Custom Fields for Exposed Opcodes 5
Migrating Unexposed BRM Opcodes for XML-Element Services 5
Migrating Unexposed BRM Opcodes for XML-String Services 7
Migrating Custom Opcodes for XML-Element Services 7
Migrating Custom Opcodes for XML-String Services 8

A Configuring Oracle Access Management for WSM OAuth 2.0

Preparing Oracle Access Management A-1
Creating an OAuth Identity Domain A-1

Copyright © 2017, 2026, Oracle and/or its affiliates.

Creating a Resource Server A-2
Creating an OAuth Client A-2
Validating Your OAuth Setup A-3
Retrieving the Token-Signing Certificate A-3

Copyright © 2017, 2026, Oracle and/or its affiliates.

About This Content

This guide provides guidelines for installing and setting up Oracle Communications Billing and Revenue Management (BRM) Web Services Manager.

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.

Copyright © 2017, 2026, Oracle and/or its affiliates.

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
- Example of Reading Account Fields in BRM Using Web Services
- Using Metrics and Tracing

About WSDL Files and BRM Opcodes

Web Services Manager exposes BRM opcodes as operations through different web services. The web services define the available opcodes and the attributes to include in each request.

The web service APIs are grouped by functional area. For example, the **BRMBillService** web service defines the billing web service APIs, and the **BRMPymtService** web service defines the payment web service APIs. Web Services Manager publishes a WSDL contract for each configured SOAP endpoint.

Web Services Manager contains two types of WSDL files: one for XML-string payloads and one for XML-element payloads. Both payload types are available in SOAP 1.1 and SOAP 1.2 variants.

The WSDL service name and endpoint path depend on the payload type and SOAP version:

- **XML-string payloads**
  - SOAP 1.1: The WSDL service name uses the standard **Service** suffix, for example, **BRMBalService**.
  - SOAP 1.2: Add **12** immediately before **Service**, for example, **BRMBal12Service**.
  - XML-string services use the **/metro/jaxws_str/** endpoint path.
- **XML-element payloads**
  - SOAP 1.1: The WSDL service name ends in **_v2**, for example, **BRMBalServices_v2**.
  - SOAP 1.2: Add **12** immediately before **_v2**, for example, **BRMBalServices12_v2**.
  - XML-element services use the **/metro/jaxws/** endpoint path.

For web services that use an XML-element payload, the WSDL describes the input in a well-defined structure. A standards-compliant SOAP development application can use the WSDL to generate a client stub.

Table 1-1 describes the SOAP 1.1 and SOAP 1.2 WSDL service names for web services that take payload as an XML string. Each row lists service names and endpoint paths in SOAP 1.1 followed by SOAP 1.2 order.

Copyright © 2017, 2026, Oracle and/or its affiliates.

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

Web Service Name	Endpoint Path	Description
BRMARService **BRMAR12Service**	**/metro/jaxws_str/BRMARService** **/metro/jaxws_str/BRMARService12**	Defines the accounts receivable web service, which includes the following opcodes: • 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 See "Accounts Receivable FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMBalService **BRMBal12Service**	**/metro/jaxws_str/BRMBalService** **/metro/jaxws_str/BRMBalService12**	Defines the balances web service, which includes the following opcodes: • 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 **Note:** You must complete the required configuration before calling PCM_OP_BAL_GET_ECE_BALANCES. See "Configuring BRM to Use PCM_OP_BAL_GET_ECE_BALANCES". See "Balance FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMBillService **BRMBill12Service**	**/metro/jaxws_str/BRMBillService** **/metro/jaxws_str/BRMBillService12**	Defines the billing web service, which includes the following opcodes: • 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 See "Billing FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMCollectionsService **BRMCollections12Service**	**/metro/jaxws_str/BRMCollectionsService** **/metro/jaxws_str/BRMCollectionsService12**	Defines the collections web service, which includes the following opcode: PCM_OP_COLLECTIONS_SET_ACTION_STATUS See "Collections Manager FM Standard Opcodes" in *BRM Opcode Guide* for more information.

Copyright © 2017, 2026, Oracle and/or its affiliates.

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

Web Service Name	Endpoint Path	Description
BRMCustcareService **BRMCustcare12Service**	**/metro/jaxws_str/BRMCustcareService** **/metro/jaxws_str/BRMCustcareService12**	Defines the customer care web service, which includes the following opcode: PCM_OP_CUSTCARE_MOVE_ACCT
BRMCustService **BRMCust12Service**	**/metro/jaxws_str/BRMCustService** **/metro/jaxws_str/BRMCustService12**	Defines the customer web service, which includes the following opcodes: • 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 See "Customer FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMInvService **BRMInv12Service**	**/metro/jaxws_str/BRMInvService** **/metro/jaxws_str/BRMInvService12**	Defines the invoicing web service, which includes the following opcode: PCM_OP_INV_VIEW_INVOICE **Note:** Configure your client application to convert the invoice data received from PCM_OP_INV_VIEW_INVOICE into the appropriate format. See "About Invoicing Output XML Data" in *BRM JCA Resource Adapter*. See "Invoicing FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMPricingService **BRMPricing12Service**	**/metro/jaxws_str/BRMPricingService** **/metro/jaxws_str/BRMPricingService12**	Defines the pricing web service, which includes the following opcodes: • PCM_OP_PRICE_COMMIT_PRODUCT • PCM_OP_PRICE_COMMIT_DISCOUNT • PCM_OP_PRICE_SET_PRICE_LIST • PCM_OP_PRICE_GET_PRICE_LIST
BRMPymtService **BRMPymt12Service**	**/metro/jaxws_str/BRMPymtService** **/metro/jaxws_str/BRMPymtService12**	Defines the payment web service, which includes the following opcode: PCM_OP_PYMT_COLLECT See "Payment FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMReadService **BRMRead12Service**	**/metro/jaxws_str/BRMReadService** **/metro/jaxws_str/BRMReadService12**	Defines the read web service, which includes the following opcodes: • PCM_OP_READ_FLDS • PCM_OP_READ_OBJ • PCM_OP_SEARCH See "LDAP Base Opcodes" in *BRM Opcode Guide* for more information.

Copyright © 2017, 2026, Oracle and/or its affiliates.

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

Web Service Name	Endpoint Path	Description
BRMSubscriptionService **BRMSubscription12Service**	**/metro/jaxws_str/BRMSubscriptionService** **/metro/jaxws_str/BRMSubscriptionService12**	Defines the subscription web service, which includes the following opcodes: • PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT • PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT • PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION • PCM_OP_SUBSCRIPTION_CHANGE_DEAL • PCM_OP_SUBSCRIPTION_ORDERED_BALGRP • PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY • 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 See "Subscription Management FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMBaseService **BRMBase12Service**	**/metro/jaxws_str/BRMBaseService** **/metro/jaxws_str/BRMBaseService12**	Defines a base web service that includes PCM_OP_READ_FLDS, PCM_OP_READ_OBJ, and PCM_OP_SEARCH, and exposes the generic **opcode** and **opcodeWithFlags** operations.

Table 1-2 describes the SOAP 1.1 and SOAP 1.2 WSDL service names for web services that take the payload as an XML element. Within each row, service names and endpoint paths are listed in SOAP 1.1 followed by SOAP 1.2 order. The Data Export service uses the same WSDL service name for both supplied WSDL versions.

Copyright © 2017, 2026, Oracle and/or its affiliates.

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

Web Service Name	Endpoint Path	Description
BRMActServices_v2 **BRMActServices12_v2**	**/metro/jaxws/BRMActService** **/metro/jaxws/BRMActService12**	Defines the activity web service, which includes the following opcodes: • PCM_OP_ACT_ACTIVITY • PCM_OP_ACT_CALC_MAX_USAGE • PCM_OP_ACT_FIND • PCM_OP_ACT_LOAD_SESSION See "Activity FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMARServices_v2 **BRMARServices12_v2**	**/metro/jaxws/BRMARService** **/metro/jaxws/BRMARService12**	Defines the accounts receivable web service, which includes the following opcodes: • 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_DETAIL • 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 See "Accounts Receivable FM Standard Opcodes" in *BRM Opcode Guide* for more information.

Copyright © 2017, 2026, Oracle and/or its affiliates.

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

Web Service Name	Endpoint Path	Description
BRMBalServices_v2 **BRMBalServices12_v2**	**/metro/jaxws/BRMBalService** **/metro/jaxws/BRMBalService12**	Defines the balances web service, which includes the following opcodes: • PCM_OP_BAL_CHANGE_VALIDITY • PCM_OP_BAL_GET_BALANCES • PCM_OP_BAL_GET_ECE_BALANCES • PCM_OP_BAL_GET_BAL_GRP_AND_SVC • PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC • PCM_OP_BAL_GET_ACCT_BILLINFO **Note:** You must complete the required configuration before calling PCM_OP_BAL_GET_ECE_BALANCES. See "Configuring BRM to Use PCM_OP_BAL_GET_ECE_BALANCES". See "Balance FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMBillServices_v2 **BRMBillServices12_v2**	**/metro/jaxws/BRMBillService** **/metro/jaxws/BRMBillService12**	Defines the billing web service, which includes the following opcodes: • 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 See "Billing FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMCollectionsServices_v2 **BRMCollectionsServices12_v2**	**/metro/jaxws/BRMCollectionsService** **/metro/jaxws/BRMCollectionsService12**	Defines the collections web service, which includes the following opcode: PCM_OP_COLLECTIONS_SET_ACTION_STATUS See "Collections Manager FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMContractServices_v2 **BRMContractServices12_v2**	**/metro/jaxws/BRMContractService** **/metro/jaxws/BRMContractService12**	Defines the contract web service, which includes the following opcodes: • PCM_OP_CONTRACT_CANCEL_CONTRACT • PCM_OP_CONTRACT_MODIFY_CONTRACT • PCM_OP_CONTRACT_PROCESS_MILESTONE • PCM_OP_CONTRACT_GET_MILESTONES • (SOAP 1.2 only) PCM_OP_CONTRACT_RENEW_CONTRACT

Copyright © 2017, 2026, Oracle and/or its affiliates.

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

Web Service Name	Endpoint Path	Description
BRMCustcareServices_v2 **BRMCustcareServices12_v2**	**/metro/jaxws/BRMCustcareService** **/metro/jaxws/BRMCustcareService12**	Defines the customer care web service, which includes the following opcode: PCM_OP_CUSTCARE_MOVE_ACCT
BRMCustServices_v2 **BRMCustServices12_v2**	**/metro/jaxws/BRMCustService** **/metro/jaxws/BRMCustService12**	Defines the customer web service, which includes the following opcodes: • 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 • PCM_OP_CUST_SET_BAL_GRP • PCM_OP_CUST_CHANGE_BUSINESS_PROFILE See "Customer FM Standard Opcodes" in *BRM Opcode Guide* for more information. • PCM_OP_CUST_POL_GET_PLANS • PCM_OP_CUST_POL_GET_DEALS • PCM_OP_CUST_POL_GET_PRODUCTS • PCM_OP_CUST_POL_READ_PLAN • PCM_OP_CUST_POL_VALID_NAMEINFO See "Customer FM Policy Opcodes" in *BRM Opcode Guide* for more information.
BRMInvServices_v2 **BRMInvServices12_v2**	**/metro/jaxws/BRMInvService** **/metro/jaxws/BRMInvService12**	Defines the invoicing web service, which includes the following opcode: PCM_OP_INV_VIEW_INVOICE **Note:** Configure your client application to convert the invoice data received from PCM_OP_INV_VIEW_INVOICE into the appropriate format. See "About Invoicing Output XML Data" in *BRM JCA Resource Adapter*. See "Invoicing FM Standard Opcodes" in *BRM Opcode Guide* for more information.

Copyright © 2017, 2026, Oracle and/or its affiliates.

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

Web Service Name	Endpoint Path	Description
BRMPricingServices_v2 **BRMPricingServices12_v2**	**/metro/jaxws/BRMPricingService** **/metro/jaxws/BRMPricingService12**	Defines the pricing web service, which includes the following opcodes: • PCM_OP_PRICE_COMMIT_PRODUCT • PCM_OP_PRICE_GET_DISCOUNT_INFO • PCM_OP_PRICE_GET_PRODUCT_INFO • PCM_OP_PRICE_COMMIT_DISCOUNT • PCM_OP_PRICE_GET_PRICE_LIST
BRMPymtServices_v2 **BRMPymtServices12_v2**	**/metro/jaxws/BRMPymtService** **/metro/jaxws/BRMPymtService12**	Defines the payment web service, which includes the following opcode: PCM_OP_PYMT_COLLECT See "Payment FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMReadServices_v2 **BRMReadServices12_v2**	**/metro/jaxws/BRMReadService** **/metro/jaxws/BRMReadService12**	Defines the read web service, which includes the following opcodes: • PCM_OP_READ_FLDS • PCM_OP_READ_OBJ • PCM_OP_SEARCH • PCM_OP_TEST_LOOPBACK See "LDAP Base Opcodes" in *BRM Opcode Guide* for more information.

Copyright © 2017, 2026, Oracle and/or its affiliates.

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

Web Service Name	Endpoint Path	Description
BRMSubscriptionServices_v2 **BRMSubscriptionServices12_v2**	**/metro/jaxws/BRMSubscriptionService** **/metro/jaxws/BRMSubscriptionService12**	Defines the subscription web service, which includes the following opcodes: • 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 • PCM_OP_SUBSCRIPTION_ORDERED_BALGRP • PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY • PCM_OP_SUBSCRIPTION_CALC_QUOTE See "Subscription Management FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMDataExportServices_v2	**/metro/jaxws/BRMDataExportServices_v2**	Defines a SOAP 1.1 proxy for the supported BI Publisher report-service operations. Configure **bip.endpoint**, **bip.userId**, and the wallet-supplied **bip.passwd** before using this service.

Copyright © 2017, 2026, Oracle and/or its affiliates.

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 entries to your Connection Manager (CM) configuration file (*BRM_home***/sys/cm/pin.conf**):

```
- cm em_group ece PCM_OP_BAL_GET_ECE_BALANCES
- cm em_pointer ece_bal ip emGateway_host emGateway_port
```

where:

- *emGateway_host* is the name or IP address of the server on which External Manager (EM) Gateway is running.
- *emGateway_port* is the number of the port through which EM Gateway connects to the host.

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

Example of Reading Account Fields in BRM Using Web Services

This section provides an example of reading selected fields from an account object using web services when OAuth 2.0 is not configured.

See "Sending SOAP Requests to BRM Web Services" for information about sending requests to Web Services Manager if OAuth 2.0 is enabled.

To read selected fields from an account object in BRM using web services, call the **pcmOpReadFlds** web service API, which maps to the PCM_OP_READ_FLDS opcode. The **pcmOpReadFlds** web service API is included in the **BRMReadServices_v2** web service, which contains web service APIs for read operations. 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 JAX-WS. See "Determining the WSDL URLs for Web Services Manager" for information about generating the WSDL URLs.

A sample URL for the **BRMReadServices_v2** web service is:

```
http://hostname:port/metro/jaxws/BRMReadService
```

> **Note**
>
> When OAuth 2.0 is enabled, clients must include a valid access token in each SOAP request. When OAuth 2.0 is disabled, Web Services Manager does not perform HTTP username and password authentication.

Determining the WSDL URLs for Web Services Manager

To find the WSDL URLs in Web Services Manager, go to the following URL:

```
https://hostname:port/metro
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

where:

- *hostname* is the name or IP address for Web Services Manager.
- *port* is the port number for Web Services Manager.

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

Sample SOAP Request and Response XML Files

The following samples pair each SOAP request for the **pcmOpReadFlds** web service API with its corresponding response. Replace *hostname*:*port* and the account POID with values for your BRM deployment. An empty **ACCOUNT_NO** element selects that field for reading. Response values vary by account and request; each response includes a generated CONTEXT_INFO/CORRELATION_ID.

XML-string Payload, SOAP 1.1 (jaxws_str)

Endpoint

https://*hostname*:*port*/metro/jaxws_str/BRMReadService**

Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:bus="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes/">
  <soapenv:Header/>
  <soapenv:Body>
    <bus:pcmOpReadFlds>
      <flags>0</flags>
      <PCM_OP_READ_FLDS_request><![CDATA[
        <PCM_OP_READ_FLDS_inputFlist>
          <POID>0.0.0.1 /account 1 0</POID>
          <ACCOUNT_NO/>
        </PCM_OP_READ_FLDS_inputFlist>
      ]]></PCM_OP_READ_FLDS_request>
    </bus:pcmOpReadFlds>
  </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:pcmOpReadFldsResponse xmlns:ns2="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes/">
      <PCM_OP_READ_FLDS_response><![CDATA[<?xml version="1.0"
encoding="UTF-8" standalone="no"?>
<brm:PCM_OP_READ_FLDS_outputFlist xmlns:brm="http://xmlns.oracle.com/BRM/
schemas/BusinessOpcodes">
  <brm:ACCOUNT_NO>ROOT</brm:ACCOUNT_NO>
  <brm:CONTEXT_INFO>
    <brm:CORRELATION_ID>generated-per-request</brm:CORRELATION_ID>
  </brm:CONTEXT_INFO>
  <brm:POID>0.0.0.1 /account 1 13</brm:POID>
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

```
</brm:PCM_OP_READ_FLDS_outputFlist>]]></PCM_OP_READ_FLDS_response>
    </ns2:pcmOpReadFldsResponse>
  </S:Body>
</S:Envelope>
```

XML-string Payload, SOAP 1.2 (jaxws_str)

Endpoint

https://*hostname*:*port*/metro/jaxws_str/BRMReadService12

Request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
               xmlns:bus="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes/">
  <soap:Header/>
  <soap:Body>
    <bus:pcmOpReadFlds>
      <flags>0</flags>
      <PCM_OP_READ_FLDS_request><![CDATA[
        <PCM_OP_READ_FLDS_inputFlist>
          <POID>0.0.0.1 /account 1 0</POID>
          <ACCOUNT_NO/>
        </PCM_OP_READ_FLDS_inputFlist>
      ]]></PCM_OP_READ_FLDS_request>
    </bus:pcmOpReadFlds>
  </soap:Body>
</soap:Envelope>
```

Response

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
  <S:Body>
    <ns2:pcmOpReadFldsResponse xmlns:ns2="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes/">
      <PCM_OP_READ_FLDS_response><![CDATA[<?xml version="1.0"
encoding="UTF-8" standalone="no"?>
<brm:PCM_OP_READ_FLDS_outputFlist xmlns:brm="http://xmlns.oracle.com/BRM/
schemas/BusinessOpcodes">
  <brm:ACCOUNT_NO>ROOT</brm:ACCOUNT_NO>
  <brm:CONTEXT_INFO>
    <brm:CORRELATION_ID>generated-per-request</brm:CORRELATION_ID>
  </brm:CONTEXT_INFO>
  <brm:POID>0.0.0.1 /account 1 13</brm:POID>
</brm:PCM_OP_READ_FLDS_outputFlist>]]></PCM_OP_READ_FLDS_response>
    </ns2:pcmOpReadFldsResponse>
  </S:Body>
</S:Envelope>
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

XML-element Payload, SOAP 1.1 (_v2)

Endpoint

https://*hostname*:*port*/metro/jaxws/BRMReadService

Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:bus="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">
  <soapenv:Header/>
  <soapenv:Body>
    <bus:pcmOpReadFldsRequest>
      <bus:PCM_OP_READ_FLDS_Request>
        <bus:flags>0</bus:flags>
        <bus:PCM_OP_READ_FLDS_inputFlist>
          <bus:POID>0.0.0.1 /account 1 0</bus:POID>
          <bus:ACCOUNT_NO/>
        </bus:PCM_OP_READ_FLDS_inputFlist>
      </bus:PCM_OP_READ_FLDS_Request>
    </bus:pcmOpReadFldsRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <brm:pcmOpReadFldsResponse xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">
      <brm:ACCOUNT_NO>ROOT</brm:ACCOUNT_NO>
      <brm:CONTEXT_INFO>
        <brm:CORRELATION_ID>generated-per-request</brm:CORRELATION_ID>
      </brm:CONTEXT_INFO>
      <brm:POID>0.0.0.1 /account 1 13</brm:POID>
    </brm:pcmOpReadFldsResponse>
  </S:Body>
</S:Envelope>
```

XML-element Payload, SOAP 1.2

Endpoint

https://*hostname*:*port*/metro/jaxws/BRMReadService12

Request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
               xmlns:bus="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">
  <soap:Header/>
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

```
  <soap:Body>
    <bus:pcmOpReadFldsRequest>
      <bus:PCM_OP_READ_FLDS_Request>
        <bus:flags>0</bus:flags>
        <bus:PCM_OP_READ_FLDS_inputFlist>
          <bus:POID>0.0.0.1 /account 1 0</bus:POID>
          <bus:ACCOUNT_NO/>
        </bus:PCM_OP_READ_FLDS_inputFlist>
      </bus:PCM_OP_READ_FLDS_Request>
    </bus:pcmOpReadFldsRequest>
  </soap:Body>
</soap:Envelope>
```

Response

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
  <S:Header/>
  <S:Body>
    <brm:pcmOpReadFldsResponse xmlns:brm="http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes">
      <brm:ACCOUNT_NO>ROOT</brm:ACCOUNT_NO>
      <brm:CONTEXT_INFO>
        <brm:CORRELATION_ID>generated-per-request</brm:CORRELATION_ID>
      </brm:CONTEXT_INFO>
      <brm:POID>0.0.0.1 /account 1 13</brm:POID>
    </brm:pcmOpReadFldsResponse>
  </S:Body>
</S:Envelope>
```

Using Metrics and Tracing

Web Services Manager includes additional tracing and metric reporting capabilities.

More information is contained in the following topics:

- Working with Metrics in Web Services Manager
- Enabling Tracing

Working with Metrics in Web Services Manager

Web Services Manager provides tracing and metric reporting capabilities. Metric reporting is enabled by default.

Web Services Manager exposes metric data in Prometheus format. Prometheus can scrape this data, and you can use compatible dashboard software to visualize the collected metrics.

The following metric scopes are available:

- **base**: Operating system or Java runtime measurements
- **vendor**: HTTP request and key performance indicator measurements provided by the underlying web server
- **application**: Metrics registered in the application code. Web Services Manager supports application metrics that track requests for XML-element-based invocations. Web Services

Copyright © 2017, 2026, Oracle and/or its affiliates.

Manager records successful and failed calls separately with the **status** label values **success** and **fail**. The following metrics are available:

- **http_request_duration_seconds_count**: Contains the number of recorded calls for the specified API and status.
- **http_request_duration_seconds**: Contains estimated response-duration quantiles, in seconds, for **0.5**, **0.75**, **0.95**, **0.98**, **0.99**, and **0.999**.
- **http_request_duration_seconds_max**: Contains the maximum response duration, in seconds, in the current rolling metrics window.
- **http_request_duration_seconds_sum**: Contains the sum of the recorded response durations, in seconds, for the specified API and status.

Application metrics for an individual API are included when the API is invoked through an XML-element-based web service. The **api** label identifies the API. For example:

```
http_request_duration_seconds{api="PCM_OP_READ_FLDS",status="success",quantile="0.5",scope="application"} 0.125
```

This illustrative metric indicates that, for successful PCM_OP_READ_FLDS calls in the application scope, the estimated median (0.5 quantile) response duration is 0.125 seconds.

The illustrative sample shows the minimum set of labels for this application metric. Depending on the deployment and metrics collection pipeline, additional labels can be attached to the scraped time series during service discovery, relabeling, or ingestion.

To access the metrics, use the Web Services Manager metrics URL:

https://*hostname*:*port*/observe/metrics**

To filter the metrics by scope, use the Web Services Manager metrics URL with the scope query parameter:

https://*hostname*:*port*/observe/metrics?scope=***scope***

where *scope* is one of the following: **base**, **vendor**, or **application**.

Enabling Tracing

Tracing provides visibility into Web Services Manager request processing and internal operations. This information helps you identify the source of performance issues. You can view tracing information using Zipkin, which is an open source tracing system. For more information about Zipkin and its user interface, see the Zipkin website. For more tracing configuration options, see the *Tracing Guide* in the Helidon documentation: https://helidon.io/docs/v4/se/tracing.

When tracing is enabled, the logs generated by Web Services Manager for each request contain the associated trace identifier in the **trace-id[...]** field. You can use the trace identifier to troubleshoot request-processing issues.

To enable and configure tracing:

1. Open the *BRM_home***/apps/brm_wsm/config/Infranet.properties** file.
2. Ensure that the **tracing.enabled** entry is set to **true**, which is the default:

   ```
   tracing.enabled=true
   ```

3. Set the following entries to the tracing collector's host name or IP address and listening port, respectively. Web Services Manager exports spans to the tracing collector, such as Zipkin:

Copyright © 2017, 2026, Oracle and/or its affiliates.

```
tracing.host=hostname
tracing.port=port
```

4. Save and close the file.
5. Stop and restart the Web Services Manager. See "Starting and Stopping Web Services Manager".

Copyright © 2017, 2026, Oracle and/or its affiliates.

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
- Starting and Stopping Web Services Manager
- Uninstalling Web Services Manager

Installing Web Services Manager

> **Note**
>
> If Web Services Manager is already installed, you must uninstall its features before you reinstall them.

Install the required BRM server components before or at the same time as you install and configure Web Services Manager. The required components include the Connection Manager (CM) and Data Manager (DM). See "Installing BRM" in *BRM Installation Guide* for more information.

> **Note**
>
> Oracle recommends that you install Web Services Manager on the same system as BRM.

To install Web Services Manager, see "Installing Individual BRM Components" in *BRM Installation Guide*. In the Available Product Components window, select **Webservice Manager** *Version*, where *Version* matches the BRM version that you are installing.

Starting and Stopping Web Services Manager

After you install Web Services Manager, you can use the provided scripts to start and stop the application.

From the *BRM_home***/bin** directory, run the following script to start Web Services Manager:

```
start_brm_wsm
```

When you start Web Services Manager with this script, the application uses the values in the *BRM_home***/apps/brm_wsm/config/Infranet.properties** file.

From the *BRM_home***/bin** directory, run the following script to stop Web Services Manager:

Copyright © 2017, 2026, Oracle and/or its affiliates.

```
stop_brm_wsm
```

Uninstalling Web Services Manager

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

Copyright © 2017, 2026, Oracle and/or its affiliates.

3

Configuring Web Services Manager

Learn how to configure Oracle Communications Billing and Revenue Management (BRM) Web Services Manager by reviewing its configuration reference, validating XML payloads, connecting Web Services Manager to Connection Manager (CM), restricting stored searches, configuring Java logging, and enabling HTTPS for SOAP client connections.

Topics in this document:

- Web Services Manager Configuration Reference
- Validating Input and Output XML Data
- Connecting Web Services Manager to Connection Manager
- Restricting Stored Search Templates
- Configuring BI Publisher for Data Export
- Configuring Web Services Manager Logging
- Configuring HTTPS Between SOAP Clients and Web Services Manager

Web Services Manager Configuration Reference

Web Services Manager reads its application settings from *BRM_home*/**apps/brm_wsm/config/Infranet.properties** and its Java logging settings from *BRM_home*/**apps/brm_wsm/config/logging.properties**. The start script passes the installed **Infranet.properties** path to the application. When you start Web Services Manager directly, it first looks for an explicitly supplied path. If you do not supply a path, Web Services Manager looks for **Infranet.properties** in your home directory and then in the current directory.

Environment variables and Java system properties are also configuration sources. Restart Web Services Manager after changing these settings unless the referenced guidance states otherwise.

Table 3-1 lists the supported Web Services Manager configuration options. Java PCM settings are included where they are important to a Web Services Manager deployment, but their detailed semantics and tuning belong in the Java PCM and BRM administration documentation.

> **Note**
>
> In the **Applies to** column, **Both** means the standard BRM opcode services in the **jaxws_str** and **jaxws** paths. The Data Export service is identified separately where applicable.

Copyright © 2017, 2026, Oracle and/or its affiliates.

Table 3-1 Web Services Manager Configuration Options

Configuration	Default	Purpose	Applies to	Detailed Guidance
Configuration-file argument, environment variables, and Java system properties	*BRM_home*/apps/brm_wsm/config/Infranet.properties	Selects the Web Services Manager application configuration and allows deployment-specific overrides.	Web Services Manager server-wide	See "Starting and Stopping Web Services Manager".
server.host **server.port**	**0.0.0.0** **8080**	Configures the default HTTP listener.	Both: **jaxws_str** (XML-string) and **jaxws** (XML-element)	See "Configuring HTTPS Between SOAP Clients and Web Services Manager".
server.sockets.https.port **server.sockets.https.host** **server.sockets.https.tls.enabled** **server.sockets.https.tls.endpoint-identification-algorithm** **server.sockets.https.tls.client-auth** **server.sockets.https.tls.private-key.keystore.passphrase** **server.sockets.https.tls.private-key.keystore.resource.path** **server.sockets.https.tls.trust.keystore.trust-store** **server.sockets.https.tls.trust.keystore.passphrase** **server.sockets.https.tls.trust.keystore.resource.path**	Not applicable	Configures the named HTTPS listener, server identity, client-authentication mode, and optional trust store.	Both: **jaxws_str** (XML-string) and **jaxws** (XML-element)	See "Configuring HTTPS Between SOAP Clients and Web Services Manager".
tracing.service **tracing.enabled** **tracing.protocol** **tracing.host** **tracing.port** **tracing.path** **tracing.api-version**	**brm-wsm** **true** **http** **localhost** **9411** **/api/v2/spans** **2**	Enables tracing and configures span export to the tracing collector.	Both: **jaxws_str** (XML-string) and **jaxws** (XML-element), excluding the Data Export proxy	See "Using Metrics and Tracing".

Copyright © 2017, 2026, Oracle and/or its affiliates.

Table 3-1 (Cont.) Web Services Manager Configuration Options

Configuration	Default	Purpose	Applies to	Detailed Guidance
infranet.isOAuth **infranet.certificatePath**	**true** Empty string	Enables OAuth access-token validation and identifies the PEM X.509 certificate containing the RSA public key used to verify token signatures.	Both: **jaxws_str** (XML-string) and **jaxws** (XML-element) opcode services	See "Setting Up Web Services Manager with OAuth 2.0".
infranet.connectionpool.* For example: **infranet.connectionpool.minsize**, **infranet.connectionpool.maxsize**, and **infranet.connectionpool.timeout**	Not applicable	Controls the Java PCM connection pool used by Web Services Manager.	Both: **jaxws_str** (XML-string) and **jaxws** (XML-element)	See *BRM PCM Java API Reference*.
Other **infranet.*** settings For example: **infranet.connection**, **infranet.wallet.location**, **infranet.login.type**, **infranet.PcmTimeoutInMsecs**, and **infranet.pcp.ssl.***	Not applicable	Configures the shared Java PCM connection, wallet, call timeout, PCP TLS, authentication, and customization settings used by Web Services Manager.	Both: **jaxws_str** (XML-string) and **jaxws** (XML-element)	See "Connecting Web Services Manager to Connection Manager".
webservices.input.validation.enabled **webservices.soap.input.validation.reportonly** **webservices.output.validation.enabled** **webservices.soap.output.validation.reportonly**	**true** **false** **true** **false**	Controls opcode-XSD validation and whether validation errors are reported without failing processing.	**jaxws** (XML-element) only	See "Validating Input and Output XML Data".
webservices.schema.location	**${PIN_HOME}/apps/brm_wsm/schemas**	Locates opcode schemas used for XML-element validation and schema retrieval.	**jaxws** (XML-element) validation and schema retrieval	See "Validating Input and Output XML Data".
webservices.extensions.location	**${PIN_HOME}/apps/brm_wsm/brm_extensions.xml**	Locates the BRM XML conversion extension definitions.	Both: **jaxws_str** (XML-string) and **jaxws** (XML-element)	See "Customizing Web Services Manager.".

Copyright © 2017, 2026, Oracle and/or its affiliates.

Table 3-1 (Cont.) Web Services Manager Configuration Options

Configuration	Default	Purpose	Applies to	Detailed Guidance
webservices.output.namespace.prefix.soap	**S**	Sets the SOAP envelope prefix generated for XML-element and Data Export responses.	**jaxws** (XML-element) and Data Export responses	Not applicable
webservices.output.namespace.prefix.payload	**brm**	Sets the namespace prefix used by the BRM XML payload converter.	Both: **jaxws_str** (XML-string) and **jaxws** (XML-element)	Not applicable
webservices.input.date.format **webservices.output.date.format**	**yyyy-MM-dd'T'HH:mm:ss** for both	Sets the Java **SimpleDateFormat** patterns used to parse input timestamps and format output timestamps.	Both: **jaxws_str** (XML-string) and **jaxws** (XML-element)	See "**SimpleDateFormat**".
webservices.input.date.prefixunixtstamp	**false**	In the current implementation, controls whether an epoch-seconds prefix is added to an output timestamp.	Both: **jaxws_str** (XML-string) and **jaxws** (XML-element)	Not applicable
webservices.output.date.prefixunixtstamp	**false**	In the current implementation, controls whether an epoch-seconds prefix is removed while parsing an input timestamp.	Both: **jaxws_str** (XML-string) and **jaxws** (XML-element)	Not applicable
webservices.loadcustomschema	**false**	Controls whether Metro searches all matching class-path resources while resolving customized schemas.	Web Services Manager customization	See "Customizing Web Services Manager".
webservices.descriptor	**sun-jaxws.xml**	Selects the Metro endpoint deployment descriptor loaded by Web Services Manager.	Both: **jaxws_str** (XML-string) and **jaxws** (XML-element)	See "Customizing Web Services Manager".
bip.endpoint **bip.userId** Wallet entry **bip.passwd**	Not applicable	Supplies the BI Publisher endpoint and credentials required by the Data Export service.	XML-element Data Export service only	See "Configuring BI Publisher for Data Export".

Copyright © 2017, 2026, Oracle and/or its affiliates.

Table 3-1 (Cont.) Web Services Manager Configuration Options

Configuration	Default	Purpose	Applies to	Detailed Guidance
allowed.search.template.ids	Not applicable	Restricts stored-search template object IDs accepted for **PCM_OP_SEARCH**. **None** rejects all such requests.	**jaxws** (XML-element) only in the current implementation	See "Restricting Stored Search Templates".
handlers **.level** **com.oracle.communications.brm.wsm.level** **java.util.logging.FileHandler.level** **java.util.logging.FileHandler.append** **java.util.logging.FileHandler.pattern** **java.util.logging.FileHandler.formatter**	**java.util.logging.FileHandler** **WARNING** **FINEST** **INFO** **true** **brm_wsm.log** **com.oracle.communications.brm.wsm.BRMLogFormatter**	Configures Java logging output, levels, file handling, and Web Services Manager log formatting.	Web Services Manager server-wide	See "Configuring Web Services Manager Logging".
BRM_WSM_CLASSPATH_EXT	Not applicable	Adds customer-provided customization JARs to the Web Services Manager class path.	Web Services Manager startup and customization	See "Customizing Web Services Manager".
BRM_WSM_JAVA_OPTIONS	Not applicable	Supplies additional JVM options to the Web Services Manager launcher.	Web Services Manager startup	See "Starting and Stopping Web Services Manager" and the applicable Java documentation.

Validating Input and Output XML Data

For XML-element web service invocations, Web Services Manager can validate the input and output XML structure, elements, data types, and schema constraints against the target opcode XML schema.

The opcode specifications, schemas, and WSDL files are packaged with Web Services Manager. The package also includes the following items:

- The **opspec.xsd** file. Use the XSD to write opcode specifications for custom opcodes that you expose as XML-element web services.
- The **pin_opspec_to_schema_v2** utility. Use this utility to generate XML-element schema files from opcode specification files. See "Creating Opcode Specification Schema Files" for the complete procedure.

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

Copyright © 2017, 2026, Oracle and/or its affiliates.

> **Note**
>
> This procedure includes only schema-validation settings.

1. Open the *BRM_home***/apps/brm_wsm/config/Infranet.properties** file.
2. Set the following entries to **true**.

   ```
   webservices.input.validation.enabled=true
   webservices.output.validation.enabled=true
   ```

   By default, the entries are set to **true**.
3. To log a validation error without returning a SOAP fault, set the corresponding report-only entry to **true**. Web Services Manager continues processing the request.

   ```
   webservices.soap.input.validation.reportonly=true
   webservices.soap.output.validation.reportonly=true
   ```

   By default, the entries are set to **false**.
4. Set the following entry to the directory containing the opcode XSD files:

   ```
   webservices.schema.location=schemaDirectory
   ```

   where *schemaDirectory* must be accessible to the Web Services Manager process. The default is *BRM_home***/apps/brm_wsm/schemas**.
5. Store each target opcode schema as *opcodeName***.xsd** in *schemaDirectory*.
6. Save and close the file.
7. Stop and restart Web Services Manager. See "Starting and Stopping Web Services Manager".

Connecting Web Services Manager to Connection Manager

Web Services Manager acts as a Java PCM client and connects to BRM Connection Manager (CM) over PCP. Use the Java PCM documentation for the complete connection, wallet, connection-pool, and PCP TLS configuration.

Before using Web Services Manager APIs, verify the following:

- **infranet.wallet.location** points to an Oracle wallet that contains the **infranet.connection** entry.
- The BRM user identified by **infranet.connection** is authorized to execute every opcode that clients will invoke through Web Services Manager.
- If CM requires TLS, the **infranet.pcp.ssl.*** settings match the CM listener configuration and the configured PCP TLS wallet contains the certificates required to trust CM's identity. For mutual TLS, the wallet must also contain Web Services Manager's client certificate and private key, and CM must trust that client certificate.

For more information, see the following:

- "Creating Client Applications by Using Java PCM" in *BRM Developer's Guide*
- "Using Configuration Files" in *BRM System Administrator's Guide*
- **ConnectionPool** in *BRM PCM Java API Reference*.

Copyright © 2017, 2026, Oracle and/or its affiliates.

Restricting Stored Search Templates

For XML-element invocations of PCM_OP_SEARCH, Web Services Manager can restrict requests to specific stored **/search** template object IDs. This setting does not select or connect to a BRM instance. Stored-search template IDs belong to the connected BRM database, so verify that every configured ID identifies the intended /search object in that database.

To restrict stored searches:

1. Open the *BRM_home***/apps/brm_wsm/config/Infranet.properties** file.
2. Set the following entry to a comma-separated list of permitted stored-search template object IDs:

   ```
   allowed.search.template.ids=TEMPLATE_ID_1,TEMPLATE_ID_2
   ```

   When the entry is absent, Web Services Manager does not apply the allowlist restriction. Setting the value to **None** rejects all XML-element PCM_OP_SEARCH requests. For information about stored search templates, see "Searching for Objects in the BRM Database" in *BRM Developer's Guide*.
3. Save and close the file.
4. Stop and restart Web Services Manager. See "Starting and Stopping Web Services Manager".

Configuring BI Publisher for Data Export

The **BRMDataExportServices_v2** service forwards supported report operations to Oracle BI Publisher. Before using this service, configure the BI Publisher web service base URL and the credentials that Web Services Manager adds to each outbound request.

Use the following steps to configure BI Publisher for data export:

1. Ensure that the BI Publisher user is authorized to perform every report-execution, report-scheduling, report-output and job-history, and catalog-read operation that clients start through Web Services Manager.
2. Open the *BRM_home***/apps/brm_wsm/config/Infranet.properties** file.
3. Set the following entries:

   ```
   bip.endpoint=https://BIP_host:BIP_port/xmlpserver/services/
   bip.userId=BIP_user
   ```

   bip.endpoint is the BI Publisher web service base URL. Web Services Manager appends **v2/ScheduleService**, **v2/ReportService**, or **PublicReportService**, depending on the operation being invoked.
4. Store the BI Publisher user's password as **bip.passwd** in the Oracle wallet identified by **infranet.wallet.location**:

   ```
   cd BRM_home/bin
   pin_config_editor -setconf -wallet wallet_directory -parameter bip.passwd -
   pwd
   ```


Copyright © 2017, 2026, Oracle and/or its affiliates.

where *wallet_directory* is the value configured for **infranet.wallet.location**. The utility prompts for the wallet password and then for the BI Publisher password.

5. Save and close **Infranet.properties**.
6. Stop and restart Web Services Manager. See "Starting and Stopping Web Services Manager".

See the Oracle Analytics Publisher documentation for information about accessing WSDLs and default application roles and permissions. See "Using Configuration Files" in *BRM System Administrator's Guide* for more information about storing Java PCM application passwords in an Oracle wallet.

Configuring Web Services Manager Logging

Web Services Manager uses Java Util Logging (JUL) and loads its logging configuration from *BRM_home***/apps/brm_wsm/config/logging.properties**.

The delivered configuration sets the global logging level to **WARNING**, the Web Services Manager package logging level to **FINEST**, and the file-handler level to **INFO**. It appends messages to **brm_wsm.log** in the Web Services Manager log directory and formats them with **com.oracle.communications.brm.wsm.BRMLogFormatter**.

For information about changing JUL handlers, levels, formatters, and other logging configuration, see "Java Logging Overview" in *Java Platform, Standard Edition Core Libraries*.

Configuring HTTPS Between SOAP Clients and Web Services Manager

To protect SOAP requests and responses between clients and Web Services Manager, configure TLS on a Web Services Manager HTTPS listener. This is the inbound client-to-Web Services Manager connection; PCP TLS between Web Services Manager and CM is configured separately through Java PCM.

To configure HTTPS for Web Services Manager:

1. Obtain an SSL certificate and private key and store them in a PKCS12 (**.p12**) or JKS (**.jks**) keystore.
2. Open the *BRM_home***/apps/brm_wsm/config/Infranet.properties** file.
3. Configure the HTTPS listener and server keystore:

```
server.sockets.https.port=httpsPort
server.sockets.https.host=listenAddress
server.sockets.https.tls.enabled=true
server.sockets.https.tls.client-auth=NONE
server.sockets.https.tls.private-key.keystore.passphrase=keystorePassphrase
server.sockets.https.tls.private-key.keystore.resource.path=keystorePath
```

where:

- *httpsPort*: The TCP port number for the Web Services Manager HTTPS listener.
- *listenAddress*: The host name or IP address to which the Web Services Manager HTTPS listener binds.

Copyright © 2017, 2026, Oracle and/or its affiliates.

- *keystorePassphrase*: The password required to access the server keystore containing the SSL certificate and private key.
- *keystorePath*: The file-system path to the server keystore containing the SSL certificate and private key.

4. If mutual TLS is required, set the client-authentication mode and configure the trust keystore used to validate client certificates:

   ```
   server.sockets.https.tls.client-auth=REQUIRED
   server.sockets.https.tls.trust.keystore.trust-store=true
   server.sockets.https.tls.trust.keystore.passphrase=truststorePassphrase
   server.sockets.https.tls.trust.keystore.resource.path=truststorePath
   ```

   where:

   - *truststorePassphrase*: The password required to access the trust keystore used to validate client certificates for mutual TLS.
   - *truststorePath*: The file-system path to the trust keystore used to validate client certificates for mutual TLS.

5. Save and close the file.
6. Stop and restart Web Services Manager. See "Starting and Stopping Web Services Manager".
7. Call web services through the HTTPS listener by using URLs beginning with **https://***hostname***:***port***/metrol/**.

Copyright © 2017, 2026, Oracle and/or its affiliates.

4

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

About the OAuth 2.0 Authorization Framework

Web Services Manager supports OAuth access-token validation for SOAP invocations of its BRM opcode services, for both XML-string and XML-element payloads. Web Services Manager validates tokens locally and does not call a particular authorization server at run time. You can use any authorization server that issues tokens compatible with the Web Services Manager validation requirements.

Web Services Manager accepts a compact JWT signed with RS256. Configure Web Services Manager with a PEM-encoded X.509 certificate containing the RSA public key that corresponds to the private key used to sign the token. Web Services Manager uses the certificate to validate the token signature and validates the token's validity information.

Include the access token in the **HTTP Authorization: Bearer <ACCESS_TOKEN>** header of each protected SOAP invocation. For backward compatibility, Web Services Manager also accepts a SOAP header element whose local name is code. The SOAP-header token takes precedence if a request supplies both forms.

> **Note**
>
> Passing an OAuth access token in the SOAP **<code>** header is deprecated. Use the **HTTP Authorization: Bearer <ACCESS_TOKEN>** header instead. Support for the SOAP **<code>** header may be removed in a future release.

Setting Up Web Services Manager with OAuth 2.0

To set up a client application to use OAuth 2.0 access tokens with Web Services Manager:

1. Configure a compatible authorization server to issue RS256-signed JWT access tokens to the client.
2. Obtain the authorization server's token-signing X.509 certificate in PEM format. See "Obtaining the Token-Signing Certificate".
3. Make the certificate available to Web Services Manager and enable OAuth validation. See "Configuring Web Services Manager".

Copyright © 2017, 2026, Oracle and/or its affiliates.

4. Obtain an access token according to the authorization server's instructions. For service-to-service integrations, the OAuth 2.0 client credentials grant is the typical choice for obtaining the bearer access token.
5. Submit an authenticated SOAP request and verify Web Services Manager protection. See "Sending SOAP Requests to BRM Web Services".

The examples use HTTPS endpoints and placeholders in italics. Replace each placeholder with your deployment's value before running a command.

Obtaining the Token-Signing Certificate

Obtain the token-signing certificate from the authorization-server administrator or through the authorization server's supported administration interface. If the authorization server publishes a JSON Web Key Set (JWKS) containing an **x5c** value, use the X.509 certificate associated with the token-signing key. The certificate supplied to Web Services Manager must be in PEM format and contain the RSA public key corresponding to the private key used to sign the access tokens.

Whenever the authorization server changes or rotates its token-signing certificate, update the certificate file configured for Web Services Manager before tokens signed with the new key are sent to Web Services Manager. Web Services Manager reloads the certificate when the configured file changes. If the certificate location changes, update the **infranet.certificatePath** entry in the **Infranet.properties** file and restart Web Services Manager.

Configuring Web Services Manager

The default **Infranet.properties** disables OAuth validation. Enable it explicitly. If the **infranet.isOAuth** entry is absent from the file, the handler uses **true** as its fallback value.

To configure Web Services Manager to enable OAuth validation:

1. Save the authorization server's token-signing X.509 certificate in PEM format in a directory readable by WSM. The certificate must contain the RSA public key corresponding to the private key that signs the access tokens.
2. Open the *BRM_home*/**apps/brm_wsm/config/Infranet.properties** file in a text editor and edit the following parameters:

   ```
   infranet.isOAuth=true
   infranet.certificatePath=certPath/certFile.pem
   ```

   where:

   - *certPath* is the directory to which you copied the OAuth certificate.
   - *certFile* is the name of the certificate file.
3. Save the file and restart Web Services Manager.

Sending SOAP Requests to BRM Web Services

After configuring OAuth validation in Web Services Manager, obtain a compatible access token from the authorization server and include it in each protected Web Services Manager SOAP invocation. See "About WSDL Files and BRM Opcodes" for the service endpoint URLs and SOAP operation request structure.

The following SOAP 1.1 example uses the recommended HTTP Bearer header. Replace *soapEndpointURL* with the HTTPS service endpoint URL, replace *accessToken* with the access token, and insert the SOAP operation request in the Body element:

Copyright © 2017, 2026, Oracle and/or its affiliates.

```
curl -i --request POST \
  'soapEndpointURL' \
  --header 'Content-Type: text/xml;charset=UTF-8' \
  --header 'Authorization: Bearer accessToken' \
  --data '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <!-- Insert the SOAP operation request. -->
  </soapenv:Body>
</soapenv:Envelope>'
```

For compatibility with an existing client that cannot yet send the HTTP Bearer header, the following SOAP 1.1 example sends the access token through the deprecated SOAP **<code>** header. Replace *soapEndpointURL* with the HTTPS service endpoint URL, replace *accessToken* with the access token, and insert the SOAP operation request in the Body element.

```
curl -i --request POST \
  'soapEndpointURL' \
  --header 'Content-Type: text/xml;charset=UTF-8' \
  --data '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bus="http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes">
  <soapenv:Header>
    <bus:code>accessToken</bus:code>
  </soapenv:Header>
  <soapenv:Body>
    <!-- Insert the SOAP operation request. -->
  </soapenv:Body>
</soapenv:Envelope>'
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

5

Customizing Web Services Manager

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

Topics in this document:

- Setting Up Web Services Manager to Support Custom Fields in Opcodes
- Setting Up Web Services Manager to Support Unexposed Opcodes for XML-Element Services
- Setting Up Web Services Manager to Support Unexposed Opcodes for XML-String Services
- Setting Up Web Services Manager to Support Custom Opcodes
- Generating the Schema Files for Your System

For information about enabling or disabling validation of input and output XML data, see "Validating Input and Output XML Data".

Setting Up Web Services Manager to Support Custom Fields in Opcodes

You can expose custom fields in BRM opcodes that are exposed by default as web services by doing the following:

1. Add the custom field to the opcode in BRM. See "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide* for instructions.
2. Update the appropriate **.xsd** file with the new fields.
3. Copy the updated **.xsd** file to the location specified by the **webservices.schema.location** entry in the *BRM_home***/apps/brm_wsm/config/Infranet.properties** file.
4. If you want to control XML validation, see "Validating Input and Output XML Data".
5. Stop and restart Web Services Manager.

Setting Up Web Services Manager to Support Unexposed Opcodes for XML-Element Services

You can expose BRM opcodes that are not exposed by default as web services by completing this procedure.

See "About WSDL Files and BRM Opcodes" for information about the opcodes that are exposed by default.

To enable Web Services Manager to support these opcodes:

1. Generate the XSD file for your system. See "Generating the Schema Files for Your System" for instructions.

Copyright © 2017, 2026, Oracle and/or its affiliates.

2. Copy the generated XSD file to the location specified in **webservices.schema.location** in the *BRM_home***/apps/brm_wsm/config/Infranet.properties** file.
3. Create a WSDL file for the web service. See "Generating WSDL Files for Web Services" in *BRM JCA Resource Adapter*.
4. Generate the **sun-jaxws.xml** file using one of the following endpoints:
   - **/configurations/endpoints/default** (to get the default deployment descriptor)
   - **/configurations/endpoints** (to get a custom deployment descriptor, if already deployed)
5. Update the corresponding **<endpoint>** tag in the generated **sun-jaxws.xml** file for your implementation so that it points to your customized endpoint.
   - To expose the opcode under an existing service group:
     a. Update only the **wsdl** attribute to point to the path of the WSDL file, which will be placed inside a JAR file later in this procedure.

        ```
        <endpoint name="endpointName"
            implementation="pathToClass"
            url-pattern="urlPattern"
            wsdl="WEB-INF/wsdl/customDir/customWsdl"/>
        ```

        where:
        - *endpointName* is the logical name of the endpoint for the web service.
        - *pathToClass* is the fully qualified name of the implementation class.
        - *urlPattern* is the URL pattern to access the web service.
        - *customDir* is a custom directory created to avoid conflicts with the standard Web Services Manager WSDL files. Ensure that your custom WSDL is located in this directory.
        - *customWsdl* is the name of the WSDL file. Ensure that you add the same custom WSDL file to the JAR file later in this procedure.

        For example:

        ```
        <endpoint name="JAXWS_BRMCustomService"

        implementation="com.oracle.communications.brm.wsm.jaxws.services.cust.BRMCust
        omServicePttImpl"
            url-pattern="/jaxws/BRMCustomService"
            wsdl="WEB-INF/wsdl/customDir/BRMCustomServices_v2.wsdl"/>
        ```

     b. Create a JAR file that contains your WSDL file and your edited **sun-jaxws.xml** file in the following structure:
        - **WEB-INF/wsdl/***customDir***/***customWsdl*
        - *sunjaxwsPath***/sun-jaxws.xml**

        where *sunjaxwsPath* is the path to the **sun-jaxws.xml** you created during this procedure.
   - To expose the opcode under a new service group:
     a. Generate the web service implementation class. See "Generating the Web Service Implementation Class for XML Element Services".
     b. Generate the **sun-jaxws.xml** file and add the new endpoint to it. See "Configuring XML-Element Service Endpoints".

Copyright © 2017, 2026, Oracle and/or its affiliates.

6. Update the **Infranet.properties** file with the following new values:

   webservices.descriptor=*sunjaxwsPath*/**sun-jaxws.xml**
   webservices.loadcustomschema=true

7. Set the BRM_WSM_CLASSPATH_EXT environment variable to the full path of the JAR file you created.

8. Stop and restart the Web Services Manager. See "Starting and Stopping Web Services Manager".

Generating the Web Service Implementation Class for XML Element Services

To generate the web service implementation class for your XML-element services:

1. Create an implementation class for your new web service.

   The following shows a sample implementation class. Ensure that you update *packageName*, *ServiceName*, *PortName*, *ServiceClassName*, and *BindingSOAPVersion* with your custom values.

```
package PackageName;

// Copyright (c) 2024, 2026, Oracle and/or its affiliates.

import java.util.List;
import java.util.logging.Logger;
import java.util.logging.Level;
import java.util.concurrent.ConcurrentHashMap;
import java.time.Duration;

import io.helidon.service.registry.Services;
import io.helidon.metrics.api.MeterRegistry;
import io.helidon.metrics.api.Metrics;
import io.helidon.metrics.api.Timer;
import io.helidon.metrics.api.Tag;
import io.helidon.tracing.Span;

import jakarta.annotation.Resource;
import jakarta.jws.HandlerChain;
import jakarta.jws.soap.SOAPBinding;
import jakarta.xml.ws.Provider;
import jakarta.xml.ws.ServiceMode;
import jakarta.xml.ws.BindingType;
import jakarta.xml.ws.WebServiceContext;
import jakarta.xml.ws.WebServiceProvider;
import jakarta.xml.ws.soap.SOAPFaultException;
import jakarta.xml.soap.SOAPMessage;
import jakarta.xml.soap.SOAPException;
import jakarta.xml.soap.SOAPElement;
import jakarta.xml.soap.SOAPFactory;
import jakarta.xml.soap.Detail;
import jakarta.xml.soap.SOAPFault;
import jakarta.xml.soap.SOAPConstants;

import javax.xml.namespace.QName;

import org.w3c.dom.Document;

import com.portal.pcm.DeterminateException;
import com.portal.pcm.EBufException;
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

```
import com.oracle.communications.brm.wsm.jaxws.ApiRequestProcessor;
import com.oracle.communications.brm.wsm.utils.WebServicesUtilities;
import com.oracle.communications.brm.wsm.utils.TracerHandler;
import com.oracle.communications.brm.wsm.utils.SOAPFaultResponse;
import com.oracle.communications.brm.wsm.utils.ApplicationException;

@WebServiceProvider(
   targetNamespace = "http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes",
   serviceName = "ServiceName",
   portName = "PortName"
)
@ServiceMode(value = jakarta.xml.ws.Service.Mode.MESSAGE)
@SOAPBinding(
   style = SOAPBinding.Style.DOCUMENT,
   parameterStyle = SOAPBinding.ParameterStyle.BARE
)
@BindingType(jakarta.xml.ws.soap.SOAPBinding.BindingSOAPVersion)
@HandlerChain(file = "handler-chain.xml")
public class ServiceClassName implements ProviderSOAPMessage {
   private static Logger m_logger =
      Logger.getLogger(ServiceClassName.class.getName());

   private final MeterRegistry registry = Services.get(MeterRegistry.class);

   private final Tag successTag = Tag.create("status", "success");
   private final Tag failTag = Tag.create("status", "fail");

   private final ConcurrentHashMap<Integer, Timer> successResponseTimerMap =
      new ConcurrentHashMap<Integer, Timer>();
   private final ConcurrentHashMap<Integer, Timer> failedResponseTimerMap =
      new ConcurrentHashMap<Integer, Timer>();

   @Override
   public SOAPMessage invoke(SOAPMessage request) {
      m_logger.entering(
         ServiceClassName.class.getName(),
         "Starting invoke method execution for request"
      );

      Span span = TracerHandler.startSpan(
         null,
         ServiceClassName.class.getName() + "." + "invoke"
      );
      Exception spanException = null;

      String opcodeName = null;
      String soapVersion = null;
      String soapNamespaceUri = null;
      int opcodeNumber = -1;
      long startTime = System.nanoTime();
      SOAPMessage response = null;
      Timer currentTimer;

      try {
         // Determining SOAP request protocol version (1.1 or 1.2)
         soapNamespaceUri =
            request.getSOAPPart().getEnvelope().getNamespaceURI();

         if (SOAPConstants.URI_NS_SOAP_1_1_ENVELOPE.equals(soapNamespaceUri)) {
            soapVersion = SOAPConstants.SOAP_1_1_PROTOCOL;
         } else if (SOAPConstants.URI_NS_SOAP_1_2_ENVELOPE.equals(soapNamespaceUri))
{
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

```
      soapVersion = SOAPConstants.SOAP_1_2_PROTOCOL;
   } else {
      soapVersion = SOAPConstants.DEFAULT_SOAP_PROTOCOL;
      soapNamespaceUri = SOAPConstants.URI_NS_SOAP_ENVELOPE;
   }

   m_logger.log(
      Level.FINEST,
      "SOAP Protocol Version: " + soapVersion
         + ", SOAP Namespace URI: " + soapNamespaceUri
   );

   Document reqDoc =
      request.getSOAPBody().extractContentAsDocument();

   opcodeName =
      WebServicesUtilities.getOpcodeName(span.context(), reqDoc);
   opcodeNumber =
      WebServicesUtilities.getOpcode(span.context(), opcodeName);

   m_logger.log(
      Level.FINEST,
      "Opcode name obtained from request payload: " + opcodeName
   );

   if (!successResponseTimerMap.containsKey(opcodeNumber)) {
      m_logger.log(
         Level.FINEST,
         "Initializing success & failure metric timers"
      );

      Tag apiTag = Tag.create("api", opcodeName);

      successResponseTimerMap.put(
         opcodeNumber,
         registry.getOrCreate(
            Timer.builder("http_request_duration")
               .tags(List.of(apiTag, successTag))
         )
      );

      failedResponseTimerMap.put(
         opcodeNumber,
         registry.getOrCreate(
            Timer.builder("http_request_duration")
               .tags(List.of(apiTag, failTag))
         )
      );
   }

   ApiRequestProcessor processor = new ApiRequestProcessor();

   response = processor.processApiRequest(
      span.context(),
      reqDoc,
      opcodeName,
      opcodeNumber,
      soapVersion,
      soapNamespaceUri
   );

   currentTimer = successResponseTimerMap.get(opcodeNumber);
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

```
            currentTimer.record(
               Duration.ofNanos(System.nanoTime() - startTime)
            );
         } catch (ApplicationException e) {
            spanException = e;

            if (opcodeNumber != -1) {
               currentTimer = failedResponseTimerMap.get(opcodeNumber);
               currentTimer.record(
                  Duration.ofNanos(System.nanoTime() - startTime)
               );
            }

            throw SOAPFaultResponse.createErrorResponse(
               e,
               soapVersion,
               soapNamespaceUri
            );
         } catch (SOAPException ex) {
            spanException = ex;

            if (opcodeNumber != -1) {
               currentTimer = failedResponseTimerMap.get(opcodeNumber);
               currentTimer.record(
                  Duration.ofNanos(System.nanoTime() - startTime)
               );
            }

            throw SOAPFaultResponse.createErrorResponse(
               new ApplicationException(
                  ex,
                  ApplicationException.SERVER_EXCEPTION
               ),
               soapVersion,
               soapNamespaceUri
            );
         } finally {
            TracerHandler.endSpan(span, spanException);
            m_logger.exiting(
               ServiceClassName.class.getName(),
               "Completed invoke method execution for request"
            );
         }

         return response;
      }
   }
```

2. Create the **BRMCustomServicePttImpl.java** file and compile it using the following command:

   javac -d . -cp ".:$PIN_HOME/jars/*" *packageName*/**BRMCustomServicePttImpl.java**

   where *packageName* is the package name configured in your implementation class.

Configuring XML-Element Service Endpoints

To configure an XML-element service endpoint and package its required implementation and WSDL files:

1. Generate the **sun-jaxws.xml** file using one of the following endpoints:

Copyright © 2017, 2026, Oracle and/or its affiliates.

- **/configurations/endpoints/default** (to get the default deployment descriptor)
- **/configurations/endpoints** (to get a custom deployment descriptor if already deployed)

2. Add the new endpoint to the generated **sun-jaxws.xml** file for your implementation:

```
<endpoint name="endpointName"
    implementation="pathToClass"
    url-pattern="urlPattern"
    wsdl="WEB-INF/wsdl/customDir/customWsdl"/>
```

where:

- *endpointName* is the logical name of the endpoint for the web service.
- *pathToClass* is the fully qualified name of the implementation class.
- *urlPattern* is the URL pattern to access the web service.
- *customDir* is a custom directory created to avoid conflicts with the standard Web Services Manager WSDL files. Ensure that your custom WSDL is located in this directory.
- *customWsdl* is the name of the WSDL file you created. Ensure that you add the same custom WSDL file to the JAR file later in this procedure.

For example:

```
<endpoint name="JAXWS_BRMCustomService"
    implementation="CustomPackage.BRMCustomServicePttImpl"
    url-pattern="/jaxws/BRMCustomService"
    wsdl="WEB-INF/wsdl/customDir/BRMCustomServices_v2.wsdl"/>
```

3. Create a JAR file that contains your implementation class, its WSDL file, and your edited **sun-jaxws.xml** file in the following structure:

- **WEB-INF/wsdl/***customDir***/***customWsdl*
- *sunjaxwsPath***/sun-jaxws.xml**
- *packageName***/BRMCustomServicePttImpl.class**

where:

- *customDir* is the custom folder used to distinguish your custom WSDL from the standard WSDL files.
- *customWsdl* is the name of your WSDL file.
- *sunjaxwsPath* is the path to the **sun-jaxws.xml** you created.
- *packageName* is the package name configured in your implementation class.

Setting Up Web Services Manager to Support Unexposed Opcodes for XML-String Services

You can expose BRM opcodes that are not exposed by default as web services by completing this procedure.

See "About WSDL Files and BRM Opcodes" for information about the opcodes that are exposed by default.

To enable Web Services Manager to support these opcodes:

Copyright © 2017, 2026, Oracle and/or its affiliates.

1. To expose your new opcodes under an existing service group:
   a. Create a service implementation class to include the opcodes that are not already exposed by extending the available implementation class.

```
package com.oracle.services;

import com.oracle.communications.brm.wsm.jaxws_str.BRMCust;
import com.oracle.communications.brm.wsm.utils.ApplicationException;
import com.oracle.communications.brm.wsm.utils.OpcodeCaller;
import com.oracle.communications.brm.wsm.utils.SOAPFaultResponse;

import jakarta.jws.HandlerChain;
import jakarta.jws.WebMethod;
import jakarta.jws.WebParam;
import jakarta.jws.WebResult;
import jakarta.jws.WebService;
import jakarta.jws.soap.SOAPBinding;
import jakarta.xml.soap.SOAPConstants;

@WebService(
   name = "ExtendedBRMCustService",
   portName = "ExtendedBRMCustService_ptt",
   targetNamespace = "http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes/"
)
@SOAPBinding(style = SOAPBinding.Style.RPC)
@HandlerChain(file = "handler-chain.xml")
public class ExtendedBRMCust extends BRMCust {

   @WebMethod(operationName = "pcmOpCustValidateCustomer")
   @WebResult(name = "PCM_OP_CUST_VALIDATE_CUSTOMER_response")
   public String validateCustomer(
      @WebParam(name = "flags") int flag,
      @WebParam(name = "PCM_OP_CUST_VALIDATE_CUSTOMER_request") String inFlist
   ) {
      OpcodeCaller opcodeCaller = new OpcodeCaller();

      try {
         return opcodeCaller.opcodeWithFlags(
            "PCM_OP_CUST_VALIDATE_CUSTOMER",
            flag,
            inFlist,
            "PCM_OP_CUST_VALIDATE_CUSTOMER.xsd"
         );
      } catch (ApplicationException ex) {
         throw SOAPFaultResponse.createErrorResponse(
            ex,
            SOAPConstants.SOAP_1_1_PROTOCOL,
            SOAPConstants.URI_NS_SOAP_1_1_ENVELOPE
         );
      }
   }
}
```

   b. Compile the implementation class using the following command:

      javac -d . -cp ".:$PIN_HOME/jars/*" *packageName*/**ExtendedBRMCust.java**

      where *packageName* is the package name configured in your implementation class.

   c. Generate the **sun-jaxws.xml** file using one of the following endpoints:

      - **/configurations/endpoints/default** (to get the default deployment descriptor)

Copyright © 2017, 2026, Oracle and/or its affiliates.

- **/configurations/endpoints** (to get a custom deployment descriptor, if already deployed)

d. Update the endpoint in the generated **sun-jaxws.xml** file for your implementation:

```
<endpoint name="endpointName"
   implementation="pathToClass"
   url-pattern="urlPattern"/>
```

where:

- *endpointName* is the logical name of the endpoint for the web service.
- *pathToClass* is the fully qualified name of the implementation class.
- *urlPattern* is the URL pattern to access the web service.

For example:

```
<endpoint
   name="JAXWS_STR_BRMCustomService"
   implementation="CustomPackage.ExtendedBRMCust"
   url-pattern="/jaxws_str/BRMCustomService"/>
```

e. Create a JAR file that contains your implementation class and your edited **sun-jaxws.xml** file in the following structure:

- *sunjaxwsPath***/sun-jaxws.xml**
- *packageName***/ExtendedBRMCust.class**

where *sunjaxwsPath* is the path to the **sun-jaxws.xml** file you created, and *packageName* is the package name configured in your implementation class.

2. To expose your new opcodes under a new service group, create a new service class and add web methods by doing the following:

- Generating the Web Service Implementation Class for XML-String Services
- Configuring XML-String Service Endpoints

3. Update the **Infranet.properties** file with the following new values:

```
webservices.descriptor=sunjaxwsPath/sun-jaxws.xml
```

where *sunjaxwsPath* is the path to the **sun-jaxws.xml** you created during this procedure.

4. Add the full path to the JAR file that you created to the BRM_WSM_CLASSPATH_EXT environment variable.

5. Stop and restart the Web Services Manager. See "Starting and Stopping Web Services Manager".

Generating the Web Service Implementation Class for XML-String Services

To generate the web service implementation class for the custom service, do the following:

1. Create an implementation class for your new web service.

The following shows a sample implementation class. Replace *packageName* with your custom value. You can add multiple web methods to support multiple opcodes.

```
package packageName;

// Copyright (c) 2025, Oracle and/or its affiliates.

import com.oracle.communications.brm.wsm.utils.ApplicationException;
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

```
import com.oracle.communications.brm.wsm.utils.OpcodeCaller;
import com.oracle.communications.brm.wsm.utils.SOAPFaultResponse;

import jakarta.jws.HandlerChain;
import jakarta.jws.WebMethod;
import jakarta.jws.WebParam;
import jakarta.jws.WebResult;
import jakarta.jws.WebService;
import jakarta.jws.soap.SOAPBinding;
import jakarta.xml.soap.SOAPConstants;

/**
 * Class that implements the OOB Infranet CUST Web Service.
 *
 * <p>This class delegates all functionality related to communicating with
 * Infranet and calling opcodes to {@code OpcodeCaller}. This allows
 * flexibility in reimplementing the web service for a different application
 * server. For example, the new implementation can call the
 * {@code OpcodeCaller} utility functions.
 *
 * <p>CUSTOM_OP_GENERIC<br>
 */
@WebService(
   name = "BRMCustomService",
   targetNamespace = "http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes/",
   portName = "BRMCustomService_ptt"
)
@SOAPBinding(style = SOAPBinding.Style.RPC)
@HandlerChain(file = "handler-chain.xml")
public class BRMCustomInfra {

   @WebMethod(operationName = "customOpGeneric")
   @WebResult(name = "CUSTOM_OP_GENERIC_response")
   public String customGeneric(
      @WebParam(name = "flags") int flag,
      @WebParam(name = "CUSTOM_OP_GENERIC_request") String inFlist
   ) {
      OpcodeCaller opcodeCaller = new OpcodeCaller();

      try {
         return opcodeCaller.opcodeWithFlags(
            "CUSTOM_OP_GENERIC",
            flag,
            inFlist,
            "CUSTOM_OP_GENERIC.xsd"
         );
      } catch (ApplicationException ex) {
         throw SOAPFaultResponse.createErrorResponse(
            ex,
            SOAPConstants.SOAP_1_1_PROTOCOL,
            SOAPConstants.URI_NS_SOAP_1_1_ENVELOPE
         );
      }
   }
}
```

2. Compile the implementation class using the following command:

   javac -d . -cp ".:$PIN_HOME/jars/*" *packageName*/**BRMCustomInfra.java**

3. Generate the **sun-jaxws.xml** file using one of the following endpoints:

   - **/configurations/endpoints/default** (to use the default deployment descriptor)

Copyright © 2017, 2026, Oracle and/or its affiliates.

- **/configurations/endpoints** (to use a custom deployment descriptor)

4. Add the new endpoint in the generated **sun-jaxws.xml** file for your implementation:

```
<endpoint name="endpointName"
    implementation="pathToClass"
    url-pattern="urlPattern"/>
```

where:

- *endpointName* is the logical name of the endpoint for the web service.
- *pathToClass* is the fully qualified name of the implementation class.
- *urlPattern* is the URL pattern to access the web service.

For example:

```
<endpoint
    name="JAXWS_STR_BRMCustomService"
    implementation="CustomPackage.BRMCustomInfra"
    url-pattern="/jaxws_str/BRMCustomService"/>
```

5. Create a JAR file that contains your implementation class and your edited **sun-jaxws.xml** file in the following structure:

- *sunjaxwsPath***/sun-jaxws.xml**
- *packageName***/BRMCustomInfra.class**

where:

- *sunjaxwsPath* is the path to the **sun-jaxws.xml** you created.
- *packageName* is the package name configured in your implementation class.

Configuring XML-String Service Endpoints

To configure an XML-string service endpoint and package its required implementation and WSDL files:

1. Generate the **sun-jaxws.xml** file using one of the following endpoints:

- **/configurations/endpoints/default** (to get the default deployment descriptor)
- **/configurations/endpoints** (to get a custom deployment descriptor if already deployed)

2. Add the new endpoint to the generated **sun-jaxws.xml** file for your implementation:

```
<endpoint name="endpointName"
    implementation="pathToClass"
    url-pattern="urlPattern"/>
```

where:

- *endpointName* is the logical name of the new endpoint for the web service.
- *pathToClass* is the fully qualified name of the implementation class.
- *urlPattern* is the URL pattern to access the web service.

For example:

```
<endpoint
    name="JAXWS_STR_BRMCustomService"
    implementation="CustomPackage.BRMCustomInfra"
    url-pattern="/jaxws_str/BRMCustomService" />
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

3. Create a JAR file that contains your implementation class and your edited **sun-jaxws.xml** file in the following structure:
   - *sunjaxwsPath***/sun-jaxws.xml**
   - *packageName***/BRMCustomInfra.class**

   where:
   - *sunjaxwsPath* is the path to the **sun-jaxws.xml** you created during this procedure.
   - *packageName* is the package name configured in your implementation class.

Setting Up Web Services Manager to Support Custom Opcodes

You can expose your custom opcodes as web services by following the procedures below.

See "About WSDL Files and BRM Opcodes" for more information about the opcodes that are exposed by default.

See the following sections for details of how to support custom opcodes:

- Supporting Custom Opcodes for XML-Element Services
- Supporting Custom Opcodes for XML-String Services

Supporting Custom Opcodes for XML-Element Services

To enable Web Services Manager to support these opcodes for web services that expect an XML element payload:

1. Generate the XSD file for your system. See "Generating the Schema Files for Your System" for instructions.
2. Copy the generated XSD file to the location specified in **webservices.schema.location** in the *BRM_home***/apps/brm_wsm/config/Infranet.properties** file.
3. Create a WSDL file for the web service. See "Generating WSDL Files for Web Services" in *BRM JCA Resource Adapter*.
4. Generate the **CustomFields.jar** file. See "Making Custom Fields Available to Web Services Manager".
5. Generate the web service implementation class. See "Generating the Web Service Implementation Class for XML Element Services".
6. Generate the **sun-jaxws.xml** file and add the new endpoint to it. See "Configuring XML-Element Service Endpoints".
7. Update the **Infranet.properties** file with the following new values:

   ```
   webservices.descriptor=sunjaxwsPath/sun-jaxws.xml
   webservices.loadcustomschema=true
   infranet.custom.field.package=CustomOp_package
   ```

   where:
   - *sunjaxwsPath* is the path to the **sun-jaxws.xml** you created during this procedure.
   - *package* is the Java package name that contains the **CustomOp** class. For example, com.portal.classFiles.
8. Update the BRM_WSM_CLASSPATH_EXT environment variable to include the full paths to both **CustomFields.jar** and the JAR file you created in step 6.

Copyright © 2017, 2026, Oracle and/or its affiliates.

9. Stop and restart the Web Services Manager. See "Starting and Stopping Web Services Manager".

Supporting Custom Opcodes for XML-String Services

To enable Web Services Manager to support these opcodes for the web services that expect an XML string payload:

1. Generate the **CustomFields.jar** file. See "Making Custom Fields Available to Web Services Manager".
2. See "Generating the Web Service Implementation Class for XML-String Services".
3. See "Configuring XML-String Service Endpoints".
4. Update the **Infranet.properties** file with the following new values:

   ```
   webservices.descriptor=sunjaxwsPath/sun-jaxws.xml
   ```

   where *sunjaxwsPath* is the path to the **sun-jaxws.xml** you created during this procedure.
5. Add the full paths of the **CustomFields.jar** file and the JAR file you created to the BRM_WSM_CLASSPATH_EXT environment variable.
6. Stop and restart the Web Services Manager. See "Starting and Stopping Web Services Manager" for more information.

Making Custom Fields Available to Web Services Manager

To make custom fields available to Web Services Manager:

1. Create a **CustomOp.java** file by doing one of the following:
   - Manually create the **CustomOp.java** file.
   - Run the **parse_custom_ops_fields** script 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 "parse_custom_ops_fields" in *BRM Developer's Guide*.
2. Verify that the **CustomOp.java** file contains the following:
   - The mapping between opcode names and opcode numbers 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.

Copyright © 2017, 2026, Oracle and/or its affiliates.

- 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;
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

```
    }
}
```

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:

   ```
   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 the following:

   a. Copy **CustomFields.jar** to a directory that Web Services Manager can access. In this procedure, this directory is referred to as *CustJarLoc*.

   b. Open the *BRM_home***/apps/brm_wsm/config/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. 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
      ```

   e. Save and close the file.

   f. Add the full path to *CustJarLoc***/CustomFields.jar** to the BRM_WSM_CLASSPATH_EXT environment variable.

   g. Stop and restart the Web Services Manager. See "Starting and Stopping Web Services Manager" for more information.

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*.

Copyright © 2017, 2026, Oracle and/or its affiliates.

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. Ensure that the schema-file location matches the location 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*.

Copyright © 2017, 2026, Oracle and/or its affiliates.

6

Generating the Schema for Your Opcodes

Learn how to generate the opcode schemas and flist specifications required for a default Oracle Communications Billing and Revenue Management (BRM) Web Services Manager integration.

Topics in this document:

- Generating the Schema for an Existing Opcode
- Creating Opcode Specification Schema Files
- Specifying the XSL Rules to Create the Opcode Schema

If you customize a supported opcode or add support for a new opcode, generate the required XSD schema files.

> **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 to the location specified by the **webservices.schema.location** entry in the *BRM_home*/**apps/brm_wsm/config/Infranet.properties** file, and package the customized WSDL files in the appropriate custom JAR file.

Generating the Schema for an Existing Opcode

To generate schema files for a customized opcode already supported by Web Services Manager:

1. Modify the opcode's XML specification file. By default, the opcode specification XML files are installed in the *BRM_home*/**apps/brm_wsm/opspecs** directory, where *BRM_home* is the directory in which you installed the BRM components.
2. For web services that take payloads as XML elements:
   a. Run the **pin_opspec_to_schema_v2** script. See "Creating Opcode Specification Schema Files".
   b. Copy the customized XSD files to the location specified in the **webservices.schema.location** entry of the *BRM_home*/**apps/brm_wsm/config/Infranet.properties** file.

Creating Opcode Specification Schema Files

You must create opcode flist specification files for opcodes that you customize or add to Web Services Manager. To do so:

1. Create the specification XML files according to the *BRM_home*/**xsd/opspec.xsd** file.

Copyright © 2017, 2026, Oracle and/or its affiliates.

2. For web services that take payloads as XML elements, convert the opcode flist specification XML files to XSD schema files.
   a. Specify the BRM installation directory in the **pin_opspec_to_schema_v2** script by replacing $PIN_HOME with *BRM_home*.
   b. Go to the directory where you want to save the schema file and execute the command:

      ```
      pin_opspec_to_schema_v2 input_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.

      The utility creates an *opcodeName***.xsd** file in the directory from which you run the utility.

Specifying the XSL Rules to Create the Opcode Schema

The **pin_opspec_to_schema_v2** utility uses the *BRM_home***/xsd/pin_opspec_to_schema.xsl** style sheet to generate the schema for BRM opcodes. If your opcode references custom fields, you must customize the 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*.

Copyright © 2017, 2026, Oracle and/or its affiliates.

7

Migrating to a Helidon-Based Web Services Manager

Learn how to migrate Oracle Communications Billing and Revenue Management (BRM) Web Services Manager from an application server to a standalone, Helidon-based system.

Topics in the document:

- What Changes When Migrating to Helidon-Based Web Services Manager
- Migrating OAuth 2.0 Configuration to Helidon-Based Web Services Manager
- Migrating Web Services Manager Customizations

What Changes When Migrating to Helidon-Based Web Services Manager

Migrating from Web Services Manager deployed on an application server to Helidon-based Web Services Manager includes the following changes:

- Input Requests Remain Unchanged
- Changes to Web Services Manager URLs
- Changes to Web Services Manager Error Response Format

Input Requests Remain Unchanged

When you migrate to Helidon-based Web Services Manager, you can retain the input request structure used with Web Services Manager deployed on an application server. No changes to existing input requests are required when you migrate to Helidon-based Web Services Manager.

Changes to Web Services Manager URLs

Replace the Web Services Manager URLs used by your existing clients and integrations with the corresponding Helidon-based Web Services Manager URLs.

For example, the URL for the PCM_OP_READ_OBJ opcode (XML-element service) changes as follows:

Previous Web Services Manager URL:

`https://`*`hostname`***`:`***`port`***`/BrmWebServices/BRMReadServices_v2`**

New Web Services Manager URL:

`https://`*`hostname`***`:`***`port`***`/metro/jaxws/BRMReadService`**

To view the new URLs for all Web Services Manager services, use the Web Services Manager services URL:

Copyright © 2017, 2026, Oracle and/or its affiliates.

https://*hostname*:*port*/**metro**

where *hostname* is the host name or IP address of Web Services Manager, and *port* is the configured Web Services Manager port. The default HTTP port is **8080**.

Changes to Web Services Manager Error Response Format

If your existing clients or integrations parse the SOAP fault response returned by application-server-based Web Services Manager, update them to process the Helidon-based SOAP fault structure. The following examples show the previous and new response formats for the ERR_NAP_CONNECT_FAILED error.

Previous Web Services Manager error response

The following example shows the error response from Web Services Manager deployed on an application server. Detailed diagnostic information appears in the **<faultstring>** element, and the **<detail>** element contains a **<BRMWebServiceFault>** element.

```
<S:Fault xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:ns1="http://www.w3.org/2003/05/soap-envelope">
   <faultcode>ns0:Server</faultcode>
   <faultstring>
      DeterminateError - ERR_NAP_CONNECT_FAILED
      CreateConnections Error RetryableError - ERR_NAP_CONNECT_FAILED
      PCPConnection.java:0: ErrBuf Fields:
      Error=ERR_NAP_CONNECT_FAILED    Loc=PCP
      Field=UnknownField0    Rec=0    reserved=0
      Facility=0    MessageID=0    Time=7:13 AM    Version=0    Reserved2=0
      Args=&lt;none&gt;
      Nested Error:
      &lt;none&gt;
      :0: ErrBuf Fields:
      Error=ERR_NAP_CONNECT_FAILED    Loc=Unknown Location
      Field=UnknownField0    Rec=0    reserved=0
      Facility=0    MessageID=2    Time=7:13 AM    Version=0    Reserved2=0
      Args=&lt;none&gt;
      Nested Error:
      &lt;none&gt;
   </faultstring>
   <detail>
      <BRMWebServiceFault
         xmlns="http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes">
         <faultCode>Server</faultCode>
         <faultString>Processing Failed</faultString>
      </BRMWebServiceFault>
   </detail>
</ns0:Fault>
```

Helidon-based Web Services Manager error response

The following example shows the corresponding response from Helidon-based Web Services Manager. The new response places the BRM error identifier in the **<faultstring>** element and provides **<location>**, **<field>**, **<record>**, **<reserved>**, **<facility>**, and **<messageID>** as separate elements under **<detail>**.

```
<S:Fault xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <faultcode>S:Server</faultcode>
   <faultstring>ERR_NAP_CONNECT_FAILED</faultstring>
   <detail>
      <location>Unknown Location</location>
      <field>UnknownField0</field>
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

```
      <record>0</record>
      <reserved>0</reserved>
      <facility>0</facility>
      <messageID>2</messageID>
   </detail>
</S:Fault>
```

Migrating OAuth 2.0 Configuration to Helidon-Based Web Services Manager

You can continue to use the same resource server on your identity provider that you used to secure SOAP APIs with application-server-based Web Services Manager, provided that the authorization server issues access tokens compatible with Helidon-based Web Services Manager.

Helidon-based Web Services Manager validates OAuth access tokens locally by using the token-signing certificate. The **OAuthAccessTokenUrl**, **OAuthOldOAM**, and **OAuthDomainName** properties used with WebLogic-based Web Services Manager are no longer required.

To migrate your OAuth 2.0 configuration to a Helidon-based Web Services Manager:

1. Obtain the OAuth token-signing X.509 certificate in PEM format from your existing authorization server. The certificate must contain the RSA public key corresponding to the private key used to sign RS256 JWT access tokens. Copy the certificate to a directory that Helidon-based Web Services Manager can access.

   See "Obtaining the Token-Signing Certificate". If you use Oracle Access Management (OAM), see "Retrieving the Token-Signing Certificate".
2. Open the *BRM_home***/apps/brm_wsm/config/Infranet.properties** file in a text editor.
3. Remove the following WebLogic properties from the file:
   - **OAuthAccessTokenUrl**
   - **OAuthOldOAM**
   - **OAuthDomainName**
4. Add or modify the following entries in the file:

   ```
   infranet.isOAuth=true
   infranet.certificatePath=certPath/certFile.pem
   ```

   where:
   - *certPath* is the directory containing the OAuth token-signing certificate.
   - *certFile* is the name of the certificate file.
5. Save and close the **Infranet.properties** file.
6. Restart Web Services Manager to apply the changes.

Migrating Web Services Manager Customizations

When you move to Helidon-based Web Services Manager, you can reuse some existing customization artifacts. However, you must update their deployment and configuration settings for the new environment. The required changes depend on the type of customization.

Copyright © 2017, 2026, Oracle and/or its affiliates.

Table 7-1 summarizes which customization artifacts you can reuse and which deployment configurations you must replace, add, or update for Helidon-based Web Services Manager.

Table 7-1 Reusing and Updating Web Services Manager Customizations

Customization	Reuse or Retain	Replace or Remove	Add or Update
Custom fields for exposed opcodes	Existing custom field definitions.	No replacement or removal is specified.	Update the applicable XSD file and deploy it to the location specified by **webservices.schema.location**.
Unexposed BRM opcodes for XML-element services	Existing XSD and WSDL files.	Replace the application-server-specific service deployment configuration with the Helidon-based endpoint configuration.	Generate **sun-jaxws.xml**, package the required artifacts in a JAR file, and configure **webservices.descriptor**, **webservices.loadcustomschema**, and BRM_WSM_CLASSPATH_EXT. If you use a new service group, generate a service implementation class.
Unexposed BRM opcodes for XML-string services	Existing BRM opcodes. For an existing service group, extend the service implementation class to include the additional opcodes.	Replace the application-server-specific endpoint deployment configuration.	Create or update the service implementation class, configure **sun-jaxws.xml**, package the required artifacts in a JAR file, and configure **webservices.descriptor** and BRM_WSM_CLASSPATH_EXT.
Custom opcodes for XML-element services	Existing XSD and WSDL files. You can reuse **CustomFields.jar** if available.	Replace the application-server-specific service deployment configuration.	Generate the service implementation class and **sun-jaxws.xml**, package the service artifacts in a JAR file, and configure **infranet.custom.field.package**, **webservices.descriptor**, **webservices.loadcustomschema**, and BRM_WSM_CLASSPATH_EXT.

Copyright © 2017, 2026, Oracle and/or its affiliates.

Table 7-1 (Cont.) Reusing and Updating Web Services Manager Customizations

Customization	Reuse or Retain	Replace or Remove	Add or Update
Custom opcodes for XML-string services	Existing BRM custom opcode and field definitions. Reuse of previous service implementation artifacts is not specified.	Replace the application-server-specific endpoint deployment configuration.	Generate **CustomFields.jar** from the existing custom opcode and field definitions. Configure the XML-string service implementation and endpoints, and package the compiled implementation class and **sun-jaxws.xml** in a JAR file. Configure **webservices.descriptor** and add the full paths of both JAR files to BRM_WSM_CLASSPATH_EXT.

For instructions on each customization type, see these topics:

- Migrating Custom Fields for Exposed Opcodes
- Migrating Unexposed BRM Opcodes for XML-Element Services
- Migrating Unexposed BRM Opcodes for XML-String Services
- Migrating Custom Opcodes for XML-Element Services
- Migrating Custom Opcodes for XML-String Services

Migrating Custom Fields for Exposed Opcodes

For custom fields on already exposed opcodes, use the custom field definitions from your existing application-server-based Web Services Manager deployment to update the applicable XSD file. For Helidon-based Web Services Manager, deploy the updated XSD file to the location specified by **webservices.schema.location**.

To migrate your custom fields for already exposed opcodes to your new Helidon-based Web Services Manager:

1. Update the appropriate XML Schema Definition (XSD) file with your custom fields.
2. Move the updated XSD file to the path specified in the **webservices.schema.location** entry in the *BRM_home***/apps/brm_wsm/config/Infranet.properties** file.
3. Stop and restart Web Services Manager.

Migrating Unexposed BRM Opcodes for XML-Element Services

You can reuse the XSD and WSDL files from your existing Web Services Manager deployment. For Helidon-based Web Services Manager, you must configure the service deployment by using **sun-jaxws.xml**. You must also package the required artifacts in a JAR file and configure the required **Infranet.properties** entries. You must then make the JAR file available through BRM_WSM_CLASSPATH_EXT.

Copyright © 2017, 2026, Oracle and/or its affiliates.

To migrate unexposed BRM opcodes for XML-element services to your new Helidon-based Web Services Manager:

1. Copy the XSD files to the directory specified by the **webservices.schema.location** property in the *BRM_home***/apps/brm_wsm/config/Infranet.properties** file.
2. Generate the **sun-jaxws.xml** file using one of the following endpoints:
   - **/configurations/endpoints/default** (to get the default deployment descriptor)
   - **/configurations/endpoints** (to get a custom deployment descriptor, if already deployed)
3. To expose BRM opcodes under an *existing* service group:
   a. Update only the **wsdl** attribute with the path to the WSDL file that you place in a JAR file later in this procedure.

      ```
      <endpoint name="endpointName"
          implementation="pathToClass"
          url-pattern="urlPattern"
          wsdl="WEB-INF/wsdl/customDir/customWsdl"/>
      ```

      where:
      - *endpointName* is the logical name of the endpoint for the web service.
      - *pathToClass* is the fully qualified name of the implementation class.
      - *urlPattern* is the URL pattern to access the web service.
      - *customDir* is a custom directory created to avoid conflicts with the standard Web Services Manager WSDL files. Ensure that your custom WSDL is located in this directory.
      - *customWsdl* is the name of the WSDL file. Place your custom WSDL in this directory.

      For example:

      ```
      <endpoint name="JAXWS_BRMCustomService"

      implementation="com.oracle.communications.brm.wsm.jaxws.services.cust.BRMCustomSe
      rvicePttImpl"
          url-pattern="/jaxws/BRMCustomService"
          wsdl="WEB-INF/wsdl/customDir/BRMCustomServices_v2.wsdl"/>
      ```

   b. Create a JAR file that contains your WSDL file and your edited **sun-jaxws.xml** file in the following structure:
      - **WEB-INF/wsdl/***customDir***/***customWsdl*
      - *sunjaxwsPath***/sun-jaxws.xml**

      where *sunjaxwsPath* is the path to the **sun-jaxws.xml** file that you created during this procedure.
4. To expose BRM opcodes under a *new* service group, create a JAR file that contains your WSDL file, the **sun-jaxws.xml** file, and the new implementation class:
   a. Generate the web service implementation class. See "Generating the Web Service Implementation Class for XML Element Services".
   b. Generate the **sun-jaxws.xml** file and add the new endpoint to it. See "Configuring XML-Element Service Endpoints".
5. Update the **Infranet.properties** file with the following new values:

Copyright © 2017, 2026, Oracle and/or its affiliates.

```
webservices.descriptor=sunjaxwsPath/sun-jaxws.xml
webservices.loadcustomschema=true
```

6. Set the BRM_WSM_CLASSPATH_EXT environment variable to the full path of the JAR file you created.
7. Stop and restart Web Services Manager.

Migrating Unexposed BRM Opcodes for XML-String Services

When you move XML-string services that expose additional BRM opcodes to Helidon-based Web Services Manager, you can retain the existing SOAP request structure.

To expose the opcodes under an existing service group, extend the existing service implementation class to include the additional opcodes. To expose the opcodes under a new service group, create a new service implementation class.

After you extend or create the service implementation class, complete these steps for Helidon-based Web Services Manager:

1. Generate the **sun-jaxws.xml** file and configure the required service endpoints in that file.
2. Package the compiled service implementation class and the **sun-jaxws.xml** file in a JAR file.
3. Update the **webservices.descriptor** entry in the **Infranet.properties** file to specify the path to the **sun-jaxws.xml** file.
4. Add the full path to the JAR file to the BRM_WSM_CLASSPATH_EXT environment variable.
5. Restart Web Services Manager to apply the changes.

See "Setting Up Web Services Manager to Support Unexposed Opcodes for XML-String Services".

Migrating Custom Opcodes for XML-Element Services

You can reuse your existing XSD and WSDL files and, if available, your existing **CustomFields.jar** file. For Helidon-based Web Services Manager, you must make the custom-fields JAR available to Web Services Manager and configure the custom-field package. You must also generate the service implementation and **sun-jaxws.xml** files and package the service artifacts in a JAR file. You must then update the Helidon-based Web Services Manager configuration.

To migrate custom opcodes for XML-element services to your new Helidon-based Web Services Manager:

1. Reuse your existing XSD and WSDL files.
2. If a **CustomFields.jar** file is available, reuse it.

   Otherwise, see "Making Custom Fields Available to Web Services Manager" to create a **CustomFields.jar** file.
3. Move the **CustomFields.jar** file to a path that is accessible by Web Services Manager.
4. Update the **Infranet.properties** file with the following new values:

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

   where *package* is the Java package name that contains the **CustomOp** class, such as com.portal.classFiles.

Copyright © 2017, 2026, Oracle and/or its affiliates.

5. Add the full path of the **CustomFields.jar** file to the BRM_WSM_CLASSPATH_EXT environment variable.
6. See "Generating the Web Service Implementation Class for XML Element Services" to create and compile the implementation file.
7. See "Configuring XML-Element Service Endpoints" to create the **sun-jaxws.xml** file. Then create a JAR file, such as **custom_services.jar**, that contains the following files:
   - Compiled implementation file (**.class** file)
   - **sun-jaxws.xml** file
   - WSDL file
8. Update the **Infranet.properties** file with the following values:

   ```
   webservices.descriptor=sunjaxwsPath/sun-jaxws.xml
   webservices.loadcustomschema=true
   ```

   where *sunjaxwsPath* is the path to the **sun-jaxws.xml** file that you created during this procedure.
9. Add the full path of the **custom_services.jar** file to the BRM_WSM_CLASSPATH_EXT environment variable.
10. Stop and restart Web Services Manager.

Migrating Custom Opcodes for XML-String Services

When you configure XML-string services that use custom opcodes in Helidon-based Web Services Manager, you can retain the existing custom opcode definitions in BRM. You can also retain the SOAP request structure used by your existing clients. However, you must configure the custom web service implementation and deploy it in the Helidon-based environment.

To configure custom opcodes for XML-string services in Helidon-based Web Services Manager, complete these steps:

1. Use your existing custom opcode and field definitions to generate the **CustomFields.jar** file. See "Making Custom Fields Available to Web Services Manager" for instructions.
2. Create and compile the web service implementation class for your custom opcodes. See "Generating the Web Service Implementation Class for XML-String Services".
3. Generate the **sun-jaxws.xml** file and configure the custom service endpoints.
4. Package the compiled implementation class and the **sun-jaxws.xml** file in a JAR file. See "Configuring XML-String Service Endpoints".
5. Update the **Infranet.properties** file with the following entry:

   ```
   webservices.descriptor=sunjaxwsPath/sun-jaxws.xml
   ```

   where *sunjaxwsPath* is the directory that contains the **sun-jaxws.xml** file.
6. Add the full paths of the **CustomFields.jar** file and the JAR file containing your custom service implementation to the BRM_WSM_CLASSPATH_EXT environment variable.
7. Restart Web Services Manager to apply the changes.

For detailed instructions, see "Supporting Custom Opcodes for XML-String Services".

Copyright © 2017, 2026, Oracle and/or its affiliates.

A

Configuring Oracle Access Management for WSM OAuth 2.0

This appendix provides an optional example of configuring Oracle Access Management (OAM) to issue OAuth access tokens compatible with Oracle Communications Billing and Revenue Management (BRM) Web Services Manager. OAM is not a required WSM runtime dependency.

Topics in this appendix:

- Preparing Oracle Access Management
- Creating an OAuth Identity Domain
- Creating a Resource Server
- Creating an OAuth Client
- Validating Your OAuth Setup
- Retrieving the Token-Signing Certificate

The examples use HTTPS endpoints and placeholders in italics. Replace each placeholder with your deployment's value before running a command.

Preparing Oracle Access Management

Install a BRM-compatible version of Oracle Access Management. For supported versions, see *BRM Compatibility Matrix*. For installation information, see *Oracle Fusion Middleware Installing and Configuring Oracle Identity and Access Management*.

The remaining procedures use *encodedAdmin*, which is the Base64 encoding of the Oracle Access Management administration username and password separated by a colon (*adminUsername*:*adminPassword*). Use an account authorized to administer OAuth artifacts.

Creating an OAuth Identity Domain

Create an OAuth identity domain under which to create the resource server and OAuth client.

To create an identity domain, submit a request to the **Add a new OAuth Identity Domain** operation in the Oracle Access Management OAuth REST API. For example:

```
curl -i --request POST \
     https://oamAdminHost:oamAdminPort/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/oauthidentitydomain
     --header "Content-Type: application/json" \
     --header "Accept: application/json" \
     --header "Authorization: Basic encodedAdmin" \
     --data '{"name": "identityDomain", "description": "WSM OAuth domain",
"tokenSettings":[ { "tokenType": "ACCESS_TOKEN", "tokenExpiry": 3600 } ] }'
```

where:

Copyright © 2017, 2026, Oracle and/or its affiliates.

- *encodedAdmin* is the Base64 encoding of the Oracle Access Management administration username and password separated by a colon (*adminUsername*:*adminPassword*).
- *oamAdminHost*:*oamAdminPort* is the host name and port for the Oracle Access Management administration server.
- *identityDomain* is the name of the Oracle Access Management identity domain that you want to create.

For more information, see "Add a new OAuth Identity Domain" in *REST API for OAuth in Oracle Access Manager*.

Creating a Resource Server

Create an OAM resource-server registration and a scope for the client token requests.

To create a resource server, submit a request to the **Add a new Resource Server** operation in the Oracle Access Management OAuth REST API. For example:

```
curl -i --request POST \
      https://oamAdminHost:oamAdminPort/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/application \
     --header "Authorization: Basic encodedAdmin" \
     --header "Content-Type: application/json" \
     --header "Accept: application/json" \
     --data '{ "name": "resourceServer", "idDomain": "identityDomain",
"description": "WSM resource registration", "scopes":
[ { "scopeName":"OAUTH1", "description":"WSM access token" } ] }'
```

where *resourceServer* is the name of the resource server that you want to create. Its qualified scope is *resourceServer*.**OAUTH1**. Use the same value when creating the client and requesting tokens.

For more information, see "Add a new Resource Server" in *REST API for OAuth in Oracle Access Manager*.

Creating an OAuth Client

To create an OAuth client, submit a request to the **Add a new OAuth Client** operation in the Oracle Access Management OAuth REST API. For example:

```
curl -i --request POST \
     https://oamAdminHost:oamAdminPort/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/client \
     --header "Authorization: Basic encodedAdmin" \
     --header "Content-Type: application/json" \
     --header "Accept: application/json" \
     --data
'{"secret":"clientSecret","id":"clientId","name":"clientName","scopes":
["resourceServer.OAUTH1"],"clientType":"CONFIDENTIAL_CLIENT","idDomain":"ident
ityDomain","description":"WSM client","grantTypes":
["CLIENT_CREDENTIALS"],"defaultScope":"resourceServer.OAUTH1"}'
```

where:

- *clientId* and *clientSecret* are the client ID and client secret.

Copyright © 2017, 2026, Oracle and/or its affiliates.

- *clientName* is the name of the OAuth client that you want to create.

For more information, see "Add a new OAuth Client" in *REST API for OAuth in Oracle Access Manager*.

Validating Your OAuth Setup

Verify token issuance and inspection in OAM before configuring the signing certificate in Web Services Manager.

1. Generate an OAuth access token by using the **Create Access Token Flow** operation:

```
curl -i --request POST \
     https://oamManagedServerHost:oamManagedServerPort/oauth2/rest/token \
     --header 'Authorization: Basic encodedAdmin' \
     --header "Content-Type: application/x-www-form-
urlencoded;charset=UTF-8" \
     --header "X-OAUTH-IDENTITY-DOMAIN-NAME: identityDomain" \
     --data-urlencode 'grant_type=CLIENT_CREDENTIALS' \
     --data-urlencode 'scope=resourceServer.OAUTH1'
```

   A successful response has status **200** and includes *accessToken* and *expiresIn*.

   For more information, see "Create Access Token Flow" in *REST API for OAuth in Oracle Access Manager*.

2. Inspect the access token by using the **Validate Access Token Flow** operation:

```
curl -i --get \
     https://oamManagedServerHost:oamManagedServerPort/oauth2/rest/token/
info
     --header 'X-OAUTH-IDENTITY-DOMAIN-NAME: identityDomain' \
     --data-urlencode 'access_token=accessToken'
```

   where *accessToken* is the access token returned in step 1. A successful response has status 200 and includes token details. Treat the token as a credential, including in command history and URL logs.

   For more information, see "Validate Access Token Flow" in *REST API for OAuth in Oracle Access Manager*.

These checks validate the OAM configuration. They do not verify WSM protection. After configuring WSM, submit an authenticated SOAP invocation as described in "Sending SOAP Requests to BRM Web Services".

Retrieving the Token-Signing Certificate

Retrieve the identity domain's public signing certificate by using the **Fetch the public certificate of a given Identity Domain** operation:

```
curl -i --get \
  'https://oamManagedServerHost:oamManagedServerPort/oauth2/rest/security' \
  --header 'Authorization: Basic encodedClient' \
  --header 'Cache-Control: no-cache' \
  --data-urlencode 'identityDomainName=identityDomain'
```


Copyright © 2017, 2026, Oracle and/or its affiliates.

where *encodedClient* is the Base64-encoded format of the client ID and client secret separated by a colon (*clientId*:*clientSecret*).

For more information, see "Fetch the public certificate of a given Identity Domain" in *REST API for OAuth in Oracle Access Manager*.

The response contains a keys array. Select the RS256 signing key and obtain the certificate from its x5c array. Save the Base64 certificate value, without JSON quotes or array delimiters, in PEM format:

```
-----BEGIN CERTIFICATE-----
base64SigningCertificate
-----END CERTIFICATE-----
```

Configure WSM with this certificate by following the instructions in "Setting Up Web Services Manager with OAuth 2.0".

Copyright © 2017, 2026, Oracle and/or its affiliates.