Oracle® Communications Billing and Revenue Management

JCA Resource Adapter

Release 12.0
E51016-02
May 2021

ORACLE

Oracle Communications Billing and Revenue Management JCA Resource Adapter, Release 12.0

E51016-02

Contents

3 Deploying and Configuring JCA Resource Adapter

4 JCA Resource Adapter Utilities

Preface

This guide describes how to use JCA Resource Adapter with Oracle Communications Billing and Revenue Management (BRM).

Audience

This guide is intended for developers and system administrators.

Documentation Accessibility

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

Access to Oracle Support

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

1

Connecting J2EE-Compliant Applications to BRM

This document describes how a J2EE-compliant application can connect to the Oracle Communications Billing and Revenue Management (BRM) system by using JCA Resource Adapter and how BRM operations are exposed through the adapter.

Before using the adapter, you should be familiar with the following:

- J2EE and J2EE Connector Architecture (JCA) 1.5 Specification.
- Web Service Invocation Framework (WSIF) and Web Services Description Language (WSDL) with JCA bindings.

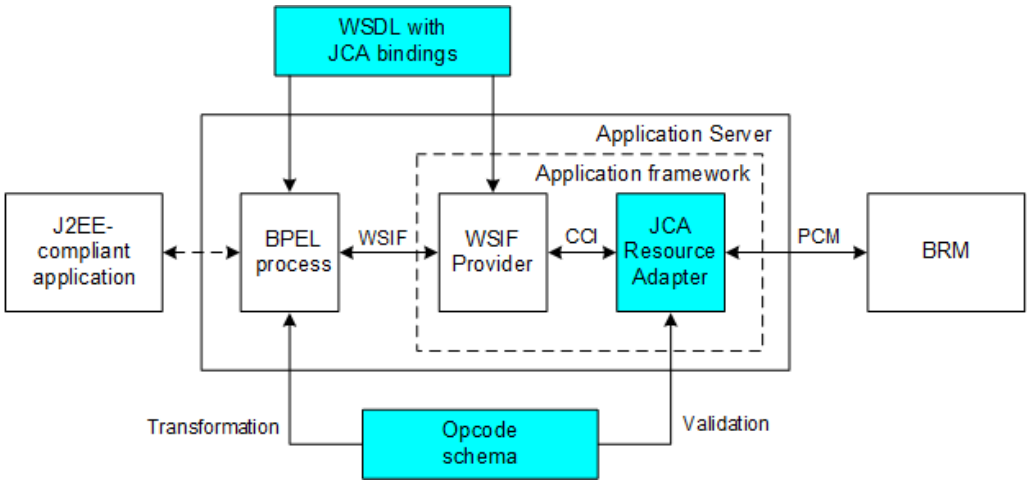
About Connecting J2EE-Compliant Applications to the BRM System

You connect J2EE-compliant external applications to your BRM system by using JCA Resource Adapter. The adapter is deployed on a J2EE-compliant application server, such as Oracle WebLogic Server. External applications send requests for information to the adapter, which then calls BRM opcodes and returns data to the external application. This enables any J2EE-compliant application to integrate with the BRM software.

For example, a customer relationship management (CRM) application, such as Oracle's Siebel CRM, that is connected to an application server can send billing and rating requests to the BRM software for processing. For information about using the adapter as part of Oracle Application Integration Architecture (Oracle AIA), see "Using BRM with Oracle Application Integration Architecture" in *BRM Developer's Guide*.

When a J2EE-compliant application sends a request to the application server, the Business Process Execution Language (BPEL) process on the application server routes the request to the adapter on the application framework (AF). Figure 1-1 illustrates the JCA Resource Adapter architecture.

Figure 1-1 JCA Resource Adapter Architecture



When the adapter receives the request, it performs the following tasks:

1. Retrieves the opcode to call from the interaction specification.
2. Optionally validates the input XML against the schema for the target opcode.
3. Converts the input XML into an input flist.
4. Calls the BRM opcode through the BRM context, using the opcode name, opcode flags, and other attributes passed in the interaction specification.

The opcode processes the request and returns the result in an output flist to the adapter. The adapter then performs the following tasks:

1. Converts the opcode's output flist into output XML and stores it in the payload of the output XML record.
2. Optionally validates the output XML against the schema for the opcode.
3. Returns the output XML to the calling application.
4. In case of an error, returns a resource exception.

The adapter connects to BRM by using internal connection pooling, which can be configured as part of the adapter's deployment when creating connection factories.

About JCA Resource Adapter

The adapter is a system-level software driver that is used by a J2EE-compliant application to connect to the BRM system. The adapter complies with J2EE Connector Architecture 1.5 and follows the guidelines for a standalone packaged adapter.
It supports synchronous outbound communication, which means that the request message originates from the J2EE-compliant application and is returned by the BRM system.

The adapter exposes the BRM API through the following:

- JCA common client interface (CCI) as an Interaction.
- WSIF provider and WSDL with JCA bindings as a WSIF API.

The adapter performs the following main functions:

- Receives data sent by the J2EE-compliant application. See "About Sending Data to JCA Resource Adapter".
- Validates the input and output XML data. See "About Validating Input and Output XML Data".
- Converts XML data into BRM flist format and calls the appropriate opcode. See "About Converting XML Data and Calling the BRM API".
- Converts output flists into XML format and returns them to the calling application. See "About Converting Output Flist Data into XML Format".

About Sending Data to JCA Resource Adapter

The adapter implements CCI as part of the client API implementation. Any J2EE-compliant application that uses CCI can send data to the adapter, but the XML data it sends must conform to the BRM opcode XML schema.

To send data to the adapter, J2EE-compliant applications send the following data in the interaction specification:

- **The BRM opcode to call**. For a list of opcodes that can be called, see the BRM WSDL files. By default, the WSDL files are installed in the *BRM_home***/apps/brm_integrations/wsdls** directory, where *BRM_home* is the directory in which the BRM server software is installed. For information about WSDL files, see "About BRM Web Services and WSDL Files".

  *BRM_home***/apps/brm_integrations/wsdls** contains sub-directories to support BPEL flows on Oracle Application Integration Architecture (AIA) for Communications. For Oracle AIA 11.*x*, use the WSDL files in *BRM_home***/apps/brm_integrations/wsdls/11g**.
- (Optional) **Opcode schema file name**. This specifies the location of the input and output schema files. By default, the schema files are installed in the *BRM_home***/apps/brm_integrations/schemas** directory.

  > **Note:**
  >
  > The schema file name is mandatory if validation is required.
- (Optional) **Opcode flags**. This defaults to zero.
- (Optional) **Whether it is a base opcode**. This specifies whether the opcode is a base opcode. See "Base Opcodes" in *BRM Developer's Guide* for more information. By default, an opcode is not a base opcode.
- (Optional) **Validation requirements**. This specifies whether the adapter validates the input XML and output XML against the opcode schema. By default, validation is turned off.

J2EE-compliant applications send the input data for the opcode in the XML record payload. This includes all data that is required by the opcode's input flist.

For more information about the interaction specification parameters, see the BRM WSDL files. For more information about the input data, see the appropriate opcode flist specification.

The adapter returns a resource exception in the following situations:

- If any of the mandatory fields are not passed in the interaction specification.
- If the input XML is not in an XML record or is not an instance of the XML record.
- If there is any error calling the opcode.
- If there is any error in the opcode's execution.
- If there is any error in the input or output validation.

About BRM Web Services and WSDL Files

The WSDL files included with the adapter define the opcodes that can be called and the attributes required to call a specific opcode. The opcodes are grouped by functional area into a Web service. For example, one Web service defines the billing opcodes and another Web service defines the payment opcodes. The adapter includes one WSDL file for each Web service.

Table 1-1 describes the WSDL files installed, by default, in the *BRM_home***/apps/brm_integrations/wsdls/10g** directory.

Table 1-1 WSDL Default Files (10g)

WSDL File Name	Description
BRMARServices.wsdl	Defines the accounts receivable Web service, which currently includes these opcodes: • PCM_OP_AR_ACCOUNT_ADJUSTMENT in • PCM_OP_AR_BILL_ADJUSTMENT • PCM_OP_AR_EVENT_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_ACTION_ITEMS • PCM_OP_AR_GET_BAL_SUMMARY • PCM_OP_AR_GET_BILLS • PCM_OP_AR_GET_BILL_ITEMS • PCM_OP_AR_ITEM_ADJUSTMENT • PCM_OP_AR_RESOURCE_AGGREGATION See "Accounts Receivable FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMBalServices.wsdl	Defines the balances Web service, which currently includes these opcodes: • PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC • PCM_OP_BAL_GET_ACCT_BILLINFO • PCM_OP_BAL_GET_BALANCES • PCM_OP_BAL_GET_BAL_GRP_AND_SVC See "Balance FM Standard Opcodes" in *BRM Opcode Guide* for more information.

Table 1-1 (Cont.) WSDL Default Files (10g)

WSDL File Name	Description
BRMBaseServices.wsdl	Defines the base Web service, which currently includes these opcodes: • PCM_OP_READ_FLDS • PCM_OP_READ_OBJ • PCM_OP_SEARCH See "Base Opcodes" in *BRM Opcode Guide* for more information.
BRMBillServices.wsdl	Defines the billing Web service, which currently includes these opcodes: • PCM_OP_BILL_DEBIT • PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT • PCM_OP_BILL_GROUP_GET_PARENT • PCM_OP_BILL_GROUP_MOVE_MEMBER • PCM_OP_BILL_MAKE_BILL_NOW See "Billing FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMCollectionsServices.wsdl	Defines the collections Web service, which currently includes this opcode: • PCM_OP_COLLECTIONS_SET_ACTION_STATUS See "Collections Manager FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMCustServices.wsdl	Defines the customer Web service, which currently includes these opcodes: • PCM_OP_CUST_COMMIT_CUSTOMER • PCM_OP_CUST_CREATE_PROFILE • PCM_OP_CUST_DELETE_PAYINFO • PCM_OP_CUST_DELETE_PROFILE • PCM_OP_CUST_MODIFY_CUSTOMER • PCM_OP_CUST_MODIFY_PROFILE • PCM_OP_CUST_SET_STATUS • PCM_OP_CUST_UPDATE_CUSTOMER • PCM_OP_CUST_UPDATE_SERVICES See "Customer FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMInvServices.wsdl	Defines the invoicing Web service, which currently includes this opcode: • PCM_OP_INV_VIEW_INVOICE **Important:** You must configure your client application to convert the invoice data received from the PCM_OP_INV_VIEW_INVOICE opcode into the appropriate format. See "About Invoicing Output XML Data". See "Invoicing FM Standard Opcodes" in *BRM Opcode Guide* for more information.
BRMPymtServices.wsdl	Defines the payment Web service, which currently includes this opcode: • PCM_OP_PYMT_COLLECT See "Payment FM Standard Opcodes" in *BRM Opcode Guide* for more information.

Table 1-1 (Cont.) WSDL Default Files (10g)

WSDL File Name	Description
BRMSubscriptionServices.wsdl	Defines the subscription Web service, which currently includes these opcodes: • PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT • PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT • PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION • PCM_OP_SUBSCRIPTION_CHANGE_DEAL • PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS • PCM_OP_SUBSCRIPTION_PURCHASE_DEAL • PCM_OP_SUBSCRIPTION_SET_BUNDLE • PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO • PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS • PCM_OP_SUBSCRIPTION_SET_PRODINFO • PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS • PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION See "Subscription Management FM Standard Opcodes" in *BRM Opcode Guide* for more information.

You can create a new WSDL file or add an opcode description to an existing WSDL file. For more information, see "Generating the WSDL Files for Your System".

About Validating Input and Output XML Data

The adapter optionally validates the input and output XML by comparing the XML fields and values against the opcode XML schema.

During the validation process, the adapter verifies that all date values passed in the input and output XML meet the ISO 8601 standard. The adapter supports the following ISO-8601 date formats:

Short Formats (without Seconds)

- yyyy-mm-ddT
- yyyy-mm-ddThh
- yyyy-mm-ddThhZ
- yyyy-mm-ddT+hh
- yyyy-mm-ddT-hh
- yyyy-mm-ddThh+hh
- yyyy-mm-ddThh-hh
- yyyy-mm-ddThhZ+hh
- yyyy-mm-ddThhZ-hh
- yyyy-mm-ddThh:mm
- yyyy-mm-ddThh:mmZ
- yyyy-mm-ddThhmm

- yyyy-mm-ddThhmmZ
- yyyy-mm-ddT+hhmm
- yyyy-mm-ddT-hhmm
- yyyy-mm-ddThh:mm+hh
- yyyy-mm-ddThh:mm-hh
- yyyy-mm-ddThh:mmZ+hh
- yyyy-mm-ddThh:mmZ-hh
- yyyy-mm-ddThhmm+hh
- yyyy-mm-ddThhmm-hh
- yyyy-mm-ddThhmmZ+hh
- yyyy-mm-ddThhmmZ-hh
- yyyy-mm-ddThh+hh:mm
- yyyy-mm-ddThh-hh:mm
- yyyy-mm-ddThhZ+hh:mm
- yyyy-mm-ddThhZ-hh:mm
- yyyy-mm-ddThh+hhmm
- yyyy-mm-ddThh-hhmm
- yyyy-mm-ddThhZ+hhmm
- yyyy-mm-ddThhZ-hhmm
- yyyy-mm-ddThh:mm+hh:mm
- yyyy-mm-ddThh:mm-hh:mm
- yyyy-mm-ddThh:mmZ+hh:mm
- yyyy-mm-ddThh:mmZ-hh:mm
- yyyy-mm-ddThh:mm+hhmm
- yyyy-mm-ddThh:mm-hhmm
- yyyy-mm-ddThh:mmZ+hhmm
- yyyy-mm-ddThh:mmZ-hhmm
- yyyy-mm-ddThhmm+hhmm
- yyyy-mm-ddThhmm-hhmm
- yyyy-mm-ddThhmmZ+hhmm
- yyyy-mm-ddThhmmZ-hhmm
- yyyy-mm-ddThhmm+hh:mm
- yyyy-mm-ddThhmm-hh:mm
- yyyy-mm-ddThhmmZ+hh:mm
- yyyy-mm-ddThhmmZ-hh:mm

Long Formats (with Seconds)

- yyyy-mm-ddThh:mm:ss
- yyyy-mm-ddThh:mm:ssZ
- yyyy-mm-ddThhmmss
- yyyy-mm-ddThhmmssZ
- yyyy-mm-ddThh:mm:ss+hh
- yyyy-mm-ddThh:mm:ss-hh
- yyyy-mm-ddThh:mm:ssZ+hh
- yyyy-mm-ddThh:mm:ssZ-hh
- yyyy-mm-ddThhmmss+hh
- yyyy-mm-ddThhmmss-hh
- yyyy-mm-ddThhmmssZ+hh
- yyyy-mm-ddThhmmssZ-hh
- yyyy-mm-ddThh:mm:ss+hh:mm
- yyyy-mm-ddThh:mm:ss-hh:mm
- yyyy-mm-ddThh:mm:ssZ+hh:mm
- yyyy-mm-ddThh:mm:ssZ-hh:mm
- yyyy-mm-ddThh:mm:ss+hhmm
- yyyy-mm-ddThh:mm:ss-hhmm
- yyyy-mm-ddThh:mm:ssZ+hhmm
- yyyy-mm-ddThh:mm:ssZ-hhmm
- yyyy-mm-ddThhmmss+hh:mm
- yyyy-mm-ddThhmmss-hh:mm
- yyyy-mm-ddThhmmssZ+hh:mm
- yyyy-mm-ddThhmmssZ-hh:mm
- yyyy-mm-ddThhmmss+hhmm
- yyyy-mm-ddThhmmss-hhmm
- yyyy-mm-ddThhmmssZ+hhmm
- yyyy-mm-ddThhmmssZ-hhmm

The adapter includes XML schema for the BRM opcodes specified in the WSDL files. If support for other opcodes, other storable classes, or subclasses is needed, you must convert them into XSD schema before you can use them with the adapter.

- For information about opcode schemas, see "About Validating the XML Schema for BRM Opcodes".
- For information about storable class schemas, see "About Validating the XML Schema for Storable Classes and Subclasses".

About Validating the XML Schema for BRM Opcodes

The adapter validates the XML data passed in by the J2EE-compliant application by using the opcode XSD schema. For example, if the input XML includes data for the PCM_OP_CUST_COMMIT_CUSTOMER opcode, the adapter validates the input XML data against the PCM_OP_CUST_COMMIT_CUSTOMER input schema. Likewise, when returning the request, the adapter validates the output XML data against the opcode's output schema.

You must generate the schema for any opcode that the adapter needs to validate by using the **pin_opspec_to_schema** utility. For more information, see "Generating the Schema for Your Opcodes".

About Validating the XML Schema for Storable Classes and Subclasses

Opcode schemas define the schema for each opcode only and contain an extension section that references storable classes. To validate flist fields that reference storable class fields, the adapter includes the schema for storable classes in the schema for the opcodes.

The opcode schemas packaged with the adapter have **include** sections for the following storable class schema files:

- **/account**
- **/billinfo**
- **/discount**
- **/event**
- **/event/billing**
- **/payinfo**
- **/product**
- **/profile**
- **/purchased_discount**
- **/purchased_product**
- **/service**

You must generate schema files for these storable classes and their subclasses. If you added any opcode schemas to the adapter, you must also generate schema files for any storable classes they need to access.

You generate the XSD schema for your storable classes and subclasses by using the **pin_dd_to_schema** utility. The utility generates the schema for all storable classes and their subclasses. For example, if you specify to generate schema for the **/service/telco/gsm** storable class, the utility generates the schema for **/service/telco/gsm**, **/service/telco/gsm/telephony**, **/service/telco/gsm/data**, and all other subclasses of the **/service/telco/gsm** storable class in your system.

To create the schema for storable classes, see "Generating the Schema for Your Storable Classes and Subclasses".

About Converting XML Data and Calling the BRM API

The adapter converts the input XML into an opcode input flist according to the rules in the XSL style sheet. The adapter then passes the flist to the appropriate opcode through the Java Portal Communications Module (PCM). If an error occurs during the opcode call or in the opcode functionality, the adapter returns an error to the calling application.

About Converting Output Flist Data into XML Format

The adapter converts output flists into XML format by using the BRM short tag name format. That is, the adapter creates XML tags by removing the “PIN_FLD_" prefix from the flist field and array name. For example, the PIN_FLD_LOGIN flist field is mapped to the LOGIN XML tag, and the PIN_FLD_SERVICES flist array is mapped to the SERVICES XML tag.

The XML tags for the PIN_FLD_INHERITED_INFO and PIN_FLD_EXTENDED_INFO substructs, however, are constructed using the following syntax:

Class`Extension`

The value of *Class* is defined in the **brm_extensions.xml** file, which specifies the flist field from which to build the tag. The file is preconfigured for the opcode and storable class schemas shipped with the adapter. If your system contains custom opcodes that reference storable class fields, you must modify this file. For information, see "Specifying the XML Tags for Extended Fields".

About Invoicing Output XML Data

When you use the PCM_OP_INV_VIEW_INVOICE opcode to retrieve and display invoices, the adapter returns an unformatted invoice to the calling application. The adapter returns invoicing data in the BUFFER XML field in hexBinary format and the type of invoice format in the TYPE_STR XML field, as described in the opcode schema. To view a formatted invoice, you must configure your client application to convert the invoicing data into the appropriate format.

About JCA Resource Adapter Connection Management

Connection management enables J2EE-compliant applications to connect to the BRM system through the adapter. The adapter manages all connections and establishes physical connections through the Java PCM.

When a J2EE-compliant application sends an XML record, the adapter performs the following tasks:

- Instantiates a CCI Connection object.
- Establishes a connection to BRM through the Java PCM.
- Stores a reference of the BRM context in the CCI Connection object.
- Returns the CCI Connection object to the application server.

About BRM JCA Resource Adapter Transaction Management

In compliance with the JCA 1.5 Specification, JCA Resource Adapter provides several levels of transaction support. You can deploy the adapter in any of the following transaction modes:

- **XA Transaction:** (Default) In this mode, the adapter supports both two-phase commit extended architecture (XA) transactions and one-phase commit transactions managed by a global transaction manager through the XAResource interface. See "About XA Transaction Support".
- **Local Transaction:** In this mode, the adapter supports only one-phase commit local transactions managed by the application server, such as Oracle WebLogic Server.
- **No Transaction:** In this mode, the transaction is managed by the application, such as Oracle AIA, that is using JCA Resource Adapter to connect to BRM.

For information about changing the transaction mode, see "Changing the JCA Resource Adapter Transaction Mode on Oracle WebLogic Server".

About XA Transaction Support

An *XA transaction* (also called a *distributed* or *global* transaction) is a group of operations implemented as a single transaction involving multiple data stores (resources) on multiple network hosts. Each data store is represented by a resource manager (such as JCA Resource Adapter), and the transaction is presided over by a global transaction manager.

To ensure that all data stores involved in an XA transaction remain synchronized, each data store must participate in a two-phase commit process:

- **Phase 1: Prepare.** After verifying whether its data store is accessible and the changes can be successfully committed, each participating resource manager votes either to commit or to roll back the transaction.
- **Phase 2: Commit.** If all participating resource managers vote to commit, the transaction manager instructs them to finalize the operation. If one or more vote to roll back, all participating resources are instructed to roll back the prepared changes.

All J2EE-compliant external applications that need XA transaction support, such as Oracle AIA, must connect to BRM through JCA Resource Adapter.

The adapter supports calls to all the methods in the J2EE standard XA API (**javax.transaction.xa.XAResource**).

For more information about XA transactions, see *XA and Oracle Controlled Distributed Transactions*.

About the XA Transaction Timeout

By default, successfully prepared XA transactions expire in BRM if a commit request is not received within a specified time after the transaction opens. The XA transaction timeout is specified in the following places, which are listed in the order of priority:

- Application server configuration settings
- Oracle DM configuration file **dm_xa_trans_timeout_in_secs** entry

Making the XA transaction timeout period long enough for the transaction manager to recover from failure and complete interrupted XA transactions without a database administrator's intervention is a good practice. See "Changing the XA Transaction Timeout Period".

The following Oracle database views contain information about expired XA transactions:

- **DBA_2PC_PENDING.** Contains all prepared XA transactions that have expired, including those that have been manually committed or rolled back but not yet purged or forgotten via an XA **forget()** request.

  This view is a static data dictionary view in the BRM database. To enable the Oracle DM process to access this view, your system administrator must grant read privileges to the BRM database user. See step 2 in "Installing JCA Resource Adapter".

  Table 1-2 describes the columns in the DBA_2PC_PENDING view.

Table 1-2 DBA_2PC_PENDING View Description

Column Name	Description
LOCAL_TRAN_ID	Specifies the branch qualifier element of an XA transaction ID (XID), which uniquely identifies the local BRM database branch of the XA transaction.
GLOBAL_TRAN_ID	Specifies the global transaction ID element of an XID, which uniquely identifies the XA transaction.
STATE	Specifies the transaction's state, which can be one of the following values: – **Collecting:** Applies only to the global coordinator or local coordinators. The node is currently collecting information from other database servers before it can decide whether it can prepare. – **Prepared:** The node has prepared and may or may not have acknowledged this to its local coordinator or transaction manager with a prepared message. However, no commit request has been received. The node remains prepared, holding any local resource locks necessary for the transaction to commit. – **Committed:** The node (any type) has committed the transaction, but other nodes involved in the transaction may not have done the same. That is, the transaction is still pending at one or more nodes. – **Forced commit:** A pending transaction can be forced to commit at the discretion of a database administrator. This entry occurs if a transaction is manually committed at a local node. – **Forced rollback:** A pending transaction can be forced to roll back at the discretion of a database administrator. This entry occurs if this transaction is manually rolled back at a local node.
MIXED	YES means that part of the transaction was committed on one node and rolled back on another node.

Table 1-2 (Cont.) DBA_2PC_PENDING View Description

Column Name	Description
TRAN_COMMIT	Specifies the transaction comment or, if transaction naming is used, the transaction name.
HOST	Specifies the host machine name.
COMMIT#	Specifies the global commit number for committed transactions.

- **DBA_PENDING_TRANSACTIONS.** Contains all prepared XA transactions waiting to be committed, whether or not they are expired. Expired transactions that have been manually committed or rolled back remain in this view until they are purged or forgotten.

Expired XA transactions can be completed by one of the following operations:

- The transaction manager's recovery process. See "About Recovering from XA Transaction Failures".
- A manual commit or rollback. See "Manually Committing or Rolling Back XA Transactions".

About Recovering from XA Transaction Failures

When the transaction manager is restarted after a failure, it sends a recover request to JCA Resource Adapter. The adapter returns a list of XA transaction branches that are in a prepared or manually completed state.

Based on information in its transaction logs, the transaction manager then issues a commit or rollback request for each transaction in the list. If a request fails with one of the following error codes (which indicate that the transaction was manually completed), the transaction manager issues a forget request to prevent the adapter from reprocessing the transaction during the next recovery process:

- PIN_ERR_XA_HEURCOM
- PIN_ERR_XA_HEURRB
- PIN_ERR_XA_HEURMIX

For more information about the XA error codes, see "BRM Error Codes" in *BRM System Administrator's Guide*.

> **Note:**
>
> To enable BRM to support the XA recovery process, you must grant read privileges to the DBA_2PC_PENDING and DBA_PENDING_TRANSACTIONS views for each BRM database user. See step 2 in "Installing JCA Resource Adapter".

About Creating Multiple Accounts in One Transaction with JCA Resource Adapter

You can create multiple BRM accounts in one transaction with JCA Resource Adapter. This is useful, for example, when a single order in Oracle Communications Order and Service Management (OSM) produces multiple new customers. It enables Oracle AIA to maintain the cross-references between the OSM order and its associated BRM accounts.

To create multiple accounts in one transaction, make all calls to the PCM_OP_CUST_COMMIT_CUSTOMER opcode any time within that transaction. The order of calling the PCM_OP_CUST_COMMIT_CUSTOMER opcode is irrelevant.

About JCA Resource Adapter Security

The adapter logs a J2EE-compliant application into the BRM software using basic security:

- **Client authentication**. The adapter authenticates a valid J2EE-compliant application with a user name and password. You can implement more robust authentication by using the adapter policies.
- **Client authorization**. By default, the adapter does not perform authorization, but you can implement it by using Oracle Identity Manager. You can also use Oracle Identity Manager to implement roles and authorization based on roles.
- **Secure communication**. A secure communication channel between the adapter and BRM is dependent on the implementation.

> **Note:**
>
> The adapter does not support reauthentication or reauthorization.

If the client passes erroneous security information, the adapter returns a security exception to indicate the error.

You set security by using entries in the adapter deployment descriptor file. For more information, see "Connecting JCA Resource Adapter to BRM from Oracle WebLogic Server ".

About JCA Resource Adapter Logging

JCA Resource Adapter supports Java Unified Logging (JUL), which enables the adapter to use the application server's logging library.

For more information, see "Configuring JCA Resource Adapter Logging on Oracle WebLogic Server".

About Deploying and Connecting JCA Resource Adapter

After installing the adapter package, you must deploy it on your application server. The connection parameters to connect the adapter to BRM are defined in the deployment descriptor packaged along with the adapter archive (RAR) file. The deployment descriptor includes the following files:

- Adapter deployment descriptor file (**ra.xml**): Includes parameters for connecting the adapter to BRM and the application server. The format of the file is defined in the J2EE Connector Architecture 1.5 Specification.
- Oracle WebLogic Server deployment descriptor file (**weblogic-ra.xml**): Defines operation parameters that are unique to Oracle WebLogic Server.

For more information, see "Deploying and Configuring JCA Resource Adapter on Oracle WebLogic Server".

About the BRMAdapterServletClient Application

Use the **BRMAdapterServletClient** application to test your custom opcodes and develop customized applications that communicate with JCA Resource Adapter.

For more information, see "Testing JCA Resource Adapter Configuration and BRM Connectivity".

2
Installing JCA Resource Adapter

This document explains how to install JCA Resource Adapter on your Oracle Communications Billing and Revenue Management (BRM) system.

System Requirements

JCA Resource Adapter is available for the Oracle Linux and Oracle Solaris operating systems.

The adapter must be deployed on a J2EE 1.4-compliant application server that has implemented the JCA 1.5 Specification. The adapter runs only in a managed environment.

Installing JCA Resource Adapter

JCA Resource Adapter is in the Web Services Manager downloadable package.

To install JCA Resource Adapter:

1. Install JCA Resource Adapter. For instructions, see "Installing Individual BRM Components" in *BRM Installation Guide*.
2. Grant read privileges to the pending extended architecture (XA) transaction database views for *each* BRM database user:
   a. Using SQL*Plus, log in to your database as the SYSDBA user:

      ```
      sqlplus sys@databaseAlias as sysdba
      ```

      where *databaseAlias* is the database alias of your BRM database.
   b. Grant read privileges to the pending XA transaction views:

      ```
      SQL> GRANT SELECT ON DBA_2PC_PENDING TO user;
      SQL> GRANT SELECT ON DBA_PENDING_TRANSACTIONS TO user;
      ```

      where *user* is the name of a BRM database user.
   c. Exit SQL*Plus.
   d. Stop and restart the BRM database to initialize the database instance with your changes.

Your JCA Resource Adapter installation is now complete.

3

Deploying and Configuring JCA Resource Adapter

This document describes how to deploy and configure JCA Resource Adapter.

Before configuring the adapter, you should be familiar with the following:

- BRM opcodes and flists. See "Understanding Flists" in *BRM Developer's Guide*.
- J2EE Connector Architecture (JCA) 1.5 Specification.
- XML, XML Schema Definition (XSD), and XML StyleSheet Language (XSL).
- Web Services Description Language (WSDL).

See also "Connecting J2EE-Compliant Applications to BRM".

Overview of the JCA Resource Adapter Configuration Procedure

The procedure for setting up JCA Resource Adapter includes the following tasks:

1. Installing the adapter on your BRM system. See "Installing JCA Resource Adapter".
2. Compiling your custom Java Portal Connection Module (PCM) classes and adding them to the adapter. See "Creating and Packaging Custom Storable Classes and Fields".
3. Generating the schema files for the adapter. See "Generating the Schema Files for Your System".
4. Specifying how to construct XML tags. See "Specifying the XML Tags for Extended Fields".
5. Generating the WSDL files for the adapter. See "Generating the WSDL Files for Your System".
6. Configuring the representation of infinite date values. See "Configuring How to Represent Infinite Date Values".
7. Deploying and connecting the adapter on your application server. See "Deploying and Configuring JCA Resource Adapter on Oracle WebLogic Server".
8. (Optional) Change the XA transaction timeout period. See "Changing the XA Transaction Timeout Period".
9. Testing JCA Resource Adapter's configuration and connectivity to BRM. See "Testing JCA Resource Adapter Configuration and BRM Connectivity".

Creating and Packaging Custom Storable Classes and Fields

JCA Resource Adapter uses Java PCM APIs to connect to BRM. If the adapter will call opcodes that reference custom storable class fields, you must make the custom storable class fields available to the adapter as follows:

- Compile your custom storable classes and add them to the adapter.
- Add the **Infranet.properites** file as a resource to the adapter. The **Infranet.properties** file defines which package contains the custom field definitions.

> **Note:**
>
> The application server class loader does not load the **Infranet.properties** file, so you must make the file available as a resource.

To make your custom and modified storable classes available to JCA Resource Adapter:

1. Use Storable Class Editor to create or modify your storable classes and fields.

   For information about editing storable classes, see "Creating, Editing, and Deleting Fields and Storable Classes" in *BRM Developer's Guide*.

2. From the **File** menu, select **Generate Custom Fields Source** to create source files for your custom fields.

   Storable Class Editor creates a C header file called **cust_flds.h**, a Java properties file called **InfranetPropertiesAdditions.properties**, and a Java source file for each custom field.

3. For each Java application that will use these fields, copy the contents of the **InfranetPropertiesAdditions.properties** file into each application's **Infranet.properties** file.

4. In the directory where Storable Class Editor created the Java source files, compile the source files:

   ```
   javac -d . *.java
   ```

5. Package the storable class files created in step 4 into a JAR file:

   ```
   jar cvf file_name.jar *.class
   ```

6. In the CLASSPATH, add the location of the JAR file.

7. Create a package that includes both the JAR file and the updated BRM **Infranet.properties** file.

8. Place the package in the adapter's working directory (*Oracle_home***/j2ee/***instance***/connectors/***adapter_deployment_name***/***adapter_deployment_name*).

9. Restart your application server.

Generating the Schema Files for Your System

JCA Resource Adapter uses schema files to validate data it sends to or receives from the BRM server.

To generate the schema files for your system:

1. If you modified any opcodes, generate schemas for the opcodes in your BRM system. See "Generating the Schema for Your Opcodes".
2. Generate schemas for the storable classes and subclasses in your BRM system. See "Generating the Schema for Your Storable Classes and Subclasses".
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".
4. Copy the opcode and storable class schema files to a location that is accessible to the adapter. Verify that this location is the same as the one 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".

Generating the Schema for Your Opcodes

The adapter package includes all the opcode schemas and flist specifications you need for a default integration.

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

The steps you must take depend on the type of changes you made, as shown in Table 3-1:

Table 3-1 Procedure for Modifying Opcodes

Opcode Modification	Procedure
Customized an opcode that JCA Resource Adapter already supports	Do the following: 1. Modify the opcode's XML specification file. By default, the opcode specification XML files are installed in the *BRM_home*/**apps/brm_integrations/opspecs** directory, where *BRM_home* is the directory in which the BRM server software is installed. See "Creating Custom Opcode Specification XML Files". 2. Run the **pin_opspec_to_schema** utility. See "Converting flist Specs into XSD Schema Files with pin_opspec_to_schema".

Table 3-1 (Cont.) Procedure for Modifying Opcodes

Opcode Modification	Procedure
Added a custom data type to an opcode that JCA Resource Adapter already supports	Do the following: 1. Modify the opcode's XML specification file. By default, the opcode specifications are installed in the *BRM_home*/**apps/brm_integrations/opspecs** directory. See "Creating Custom Opcode Specification XML Files". 2. Modify the **pin_opspec_to_schema** style sheet to handle custom data types or custom data structures. See "Specifying the XSL Rules to Create the Opcode Schema". 3. Run the **pin_opspec_to_schema** utility. See "Converting flist Specs into XSD Schema Files with pin_opspec_to_schema".
Added a custom opcode to JCA Resource Adapter	Do the following: 1. Configure BRM to recognize the custom opcode and manually create the opcode's XML specification file. See "Writing a Custom Facilities Module" in *BRM Developer's Guide*. 2. If your opcode supports custom data types or data structures, modify the **pin_opspec_to_schema** style sheet. See "Specifying the XSL Rules to Create the Opcode Schema". 3. Run the **pin_opspec_to_schema** utility. See "Converting flist Specs into XSD Schema Files with pin_opspec_to_schema". **Note:** Make sure you add your custom opcode to a WSDL file. See "Generating the WSDL Files for Your System".

Creating Custom Opcode Specification XML Files

You must create XML specification files for any opcodes that you customize or add to the adapter. Create the XML specification files by following the instructions in the *BRM_home*/**apps/brm_integrations/stylesheets/opspec.xsd** file. You can then convert the opcode specifications into XSD schema by using the **pin_opspec_to_schema** utility.

Specifying the XSL Rules to Create the Opcode Schema

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

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

Converting flist Specs into XSD Schema Files with pin_opspec_to_schema

To convert flist specifications into XSD schema files:

1. Go to the *BRM_home***/apps/brm_integrations** directory and run the following command:

   ```
   pin_opspec_to_schema -i input_file [-o output_file]
   ```

   where:

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

Generating the Schema for Your Storable Classes and Subclasses

JCA Resource Adapter uses storable class schema files to validate storable class extensions passed in the input XML. See "Connecting J2EE-Compliant Applications to BRM".

To generate storable class schema files for your system:

1. Determine which storable class files you must convert into XSD schema. See "Determining the Storable Classes to Convert into Schema".
2. Convert your storable classes and subclasses by running the **pin_dd_to_schema** utility. See "Generating Storable Class Schema Files".

Determining the Storable Classes to Convert into Schema

The adapter package includes all the storable class schema files required for a default integration. By default, they are installed in the *BRM_home***/apps/brm_integrations/schemas** directory.

You must generate additional schema files if your system contains the following:

- Extensions to the **/account**, **/billinfo**, **/discount**, **/event**, **/event/billing**, **/payinfo**, **/product**, **/profile**, **/purchased_discount**, **/purchased_product**, or **/service** storable classes. If you extended any of these storable classes, you must create a new schema file for the base storable class. For example, if you added a subclass to the **/profile** storable class, you must regenerate the **profile.xsd** schema file.
- Storable classes required by opcodes you added to the JCA Resource Adapter. If you added an opcode schema to the adapter, you can determine whether it requires a storable class by checking for an **include schemaLocation** entry at the beginning of the schema file. If the entry exists, it lists the required storable class schema files. For example, the **PCM_OP_CUST_COMMIT_CUSTOMER.xsd** schema file includes the following entries, which indicate that the opcode needs schema files for the **/service** and **/profile** storable classes:

  ```
  <?xml version = '1.0' encoding = 'UTF-8'?>
  <xsd:schema targetNamespace="http://www.xmlns.oracle.com/BRM/
  schemas/BusinessOpcodes"
  elementFormDefault="qualified"
      xmlns="http://www.xmlns.oracle.com/BRM/schemas/BusinessOpcodes"
      xmlns:op="http://www.portal.com/schemas/BusinessOpcodes"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  ```

```
<xsd:include schemaLocation="service.xsd"/>
<xsd:include schemaLocation="profile.xsd"/>

<xsd:element name="PCM_OP_CUST_COMMIT_CUSTOMER_inputFlist">
  <xsd:complexType>
```

Generating Storable Class Schema Files

JCA Resource Adapter uses the schema to validate the object fields that are passed in the input XML.

> **Note:**
>
> The utility requires a configuration file to access the BRM database. For more information, see "Using Configuration Files" in *BRM System Administrator's Guide*.

To generate the schema for the storable classes and subclasses in your system:

1. Verify that the storable class changes have been added to the BRM data dictionary.
2. Using a text editor, create a configuration file that lists all storable classes you would like to convert. For example, create a **sample.txt** file that includes the following entries:

   ```
   /service/telco/gsm
   /service/telco/gprs
   /service/sms
   /service/telephony
   /profile
   ```

3. Save and close the file.
4. (Oracle Linux only) Set the LC_ALL environment variable to C:

   ```
   setenv LC_ALL C
   ```

5. Go to the *BRM_home***/apps/brm_integrations** directory and enter the following command:

   ```
   pin_dd_to_schema  -e config_file| -r storable_class
   ```

   where:

   - *config_file* generates the schema for all storable classes listed in the configuration file and their subclasses.
   - *storable_class* generates the schema for the specified storable class and all of its subclasses.

   The utility generates the schema files in the directory from which you run the utility.

6. Copy the schema files to a location that is accessible to the adapter during the XML validation process; for example, the *BRM_home***/apps/brm_integrations/schemas**.

Specifying the Location of the Storable Class Schema Files in the Opcode Schema Files

Opcode schema files contain an **include** section that lists each storable class schema file that they need. You must manually update the opcode schema files to point to the storable class schema file's correct location.

Not all opcodes need storable classes, so only a subset of the opcode schema files have the **include** section. The following list shows the opcode schemas that are packaged with the adapter and have an **include** section:

- PCM_OP_CUST_COMMIT_CUSTOMER
- PCM_OP_CUST_CREATE_PROFILE
- PCM_OP_CUST_MODIFY_CUSTOMER
- PCM_OP_CUST_MODIFY_PROFILE
- PCM_OP_CUST_UPDATE_CUSTOMER
- PCM_OP_CUST_UPDATE_SERVICES
- PCM_OP_PYMT_COLLECT
- PCM_OP_READ_FLDS
- PCM_OP_READ_OBJ
- PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT
- PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT
- PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION
- PCM_OP_SUBSCRIPTION_CHANGE_DEAL
- PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS
- PCM_OP_SUBSCRIPTION_PURCHASE_DEAL
- PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO
- PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS
- PCM_OP_SUBSCRIPTION_SET_PRODINFO
- PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS
- PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION

Perform the following for each opcode schema that contains an **include** section for the storable class schema files:

1. Open the opcode XSD schema file in a text editor. By default, the opcode schema files are located in the *BRM_home***/apps/brm_integrations/schemas** directory.

2. Edit the **include schemaLocation** entry to point to each schema file's correct location. For example, you edit the lines shown in bold below for the **PCM_OP_CUST_COMMIT_CUSTOMER.xsd** file:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<xsd:schema targetNamespace="http://www.xmlns.oracle.com/BRM/
schemas/BusinessOpcodes"
elementFormDefault="qualified"
    xmlns="http://www.xmlns.oracle.com/BRM/schemas/BusinessOpcodes"
    xmlns:op="http://www.portal.com/schemas/BusinessOpcodes"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:include schemaLocation="service.xsd"/>
  <xsd:include schemaLocation="profile.xsd"/>

  <xsd:element name="PCM_OP_CUST_COMMIT_CUSTOMER_inputFlist">
    <xsd:complexType>
```

> **Note:**
>
> The adapter does not support accessing schemas through URLs.

3. Save and close the file.

Specifying the XML Tags for Extended Fields

The **brm_extensions.xml** file defines how to construct XML tags for PIN_FLD_INHERITED_INFO and PIN_FLD_EXTENDED_INFO substructs when converting output flists into XML. The default file is preconfigured for the opcode and storable class schemas shipped with the adapter. You must modify this file if you added to the adapter any opcodes that reference storable class fields.

The adapter creates XML tags for PIN_FLD_INHERITED_INFO and PIN_FLD_EXTENDED_INFO substructs by using the following syntax:

Class`Extension`

where *Class* is the subclass name, using an underscore (_) notation. For example, the XML tag for **/service/telco/gsm** is written as **service_telco_gsmExtension**.

To specify XML tags for extended fields:

- The flist array that contains the PIN_FLD_INHERITED_INFO or PIN_FLD_EXTENDED_INFO substruct.
- The flist field in which to look up the subclass name.

To specify XML tags for extended fields:

1. Open the *BRM_Home***/apps/brm_integrations/config/brm_extensions.xml** file in an XML editor.

> **Note:**
>
> The file is packaged with the adapter and is copied to the *Oracle_Home*/**j2ee**/*Instance*/**connectors**/*AdapterDeploymentName*/*AdapterDeploymentName* directory during the deployment process.

2. In the **opcode name** entry, add the opcode that requires the subclass fields:

   <opcode name="*Opcodename***">**

3. In the **container name** entry, add the flist field used to construct the XML tag:

   <container name="*Arrayname***">***Fieldname***</container>**

   where:

   - *Arrayname* specifies the array that contains the PIN_FLD_INHERITED_INFO or PIN_FLD_EXTENDED_INFO substruct.
   - *Fieldname* specifies the flist field in which to look up the subclass name.

   For example, the following entry for the PCM_OP_CUST_COMMIT_CUSTOMER opcode specifies that when the PIN_FLD_SERVICES output flist array contains PIN_FLD_INHERITED_INFO or PIN_FLD_EXTENDED_INFO, the adapter looks up the subclass name from the PIN_FLD_SERVICE_OBJ flist field.

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <brm_extensions>
   <opcode name="PCM_OP_CUST_COMMIT_CUSTOMER">
   <container name="PIN_FLD_SERVICES">PIN_FLD_SERVICE_OBJ</container>
   </opcode>
   ...
   </brm_extensions>
   ```

   In this example, if the PIN_FLD_SERVICE_OBJ output flist field is **0.0.0.1 / service/ip 12345 0**, *Class* is **service_ip** and the corresponding XML extension tag is **service_ipExtension**.

4. Save and close the file.
5. Restart the adapter.

Generating the WSDL Files for Your System

The adapter package includes all the WSDL files you need for a default implementation. If you customized the adapter to support additional opcodes, you must create or modify a WSDL file.

To generate a WSDL file:

- Create an XML configuration file that specifies the opcodes associated with a Web service. See "Defining a Web Service".
- Generate the WSDL file for the Web service by running the **pin_wsdl_generator** utility. See "Generating WSDL Files for Web Services".

Defining a Web Service

To define a Web service, specify the following in a configuration file:

- The Web service name
- The opcodes associated with the Web service
- The location of each opcode's schema files

You create the configuration file manually by using an XML editor. The adapter provides a sample XML configuration file (*BRM_home*/**apps/brm_integrations/config/pin_wsdl_generator.xml**) that you can use as a guide.

The configuration file includes four main sections: **namespaces**, **groups**, **binding-info**, and **service-info**. You configure the **groups** section only; the **namespaces**, **binding-info**, and **service-info** sections are preconfigured.

Defining Default Values for a Web Service

You specify the default values for a Web service in the **group-info name** entry. The default values apply to all opcodes in the Web service, but they are overridden by any opcode-specific values defined in the **opcode name** entry. For information, see "Defining Settings for Individual Opcodes".

```
<group-info name="BASE" base="True" includePath="../schemas/"
wsdlFileInFix="Base"  validation="ValidationNotRequired"
validationAttr="ValidationNotRequired">
```

where:

- **group-info name** specifies the name of the Web service.
- **base** specifies whether the opcodes in this Web service are base opcodes. See "Base Opcodes" in *BRM Developer's Guide*. The default is **False**.
- **includePath** specifies the relative path to the schema files. The utility adds this value to the WSDL file's **include** entries.
- **wsdlFileInFix** specifies the value to add to the WSDL file name, using the following syntax: **BRM***value***Services.wsdl**. For example, if **wsdlFileInFix** is set to **Bill**, the WSDL file name is **BRMBillServices.wsdl**. If this attribute is not present, the utility uses the **group-info name** value in the file name's syntax.
- **validation** specifies whether the adapter validates the input and output XML for the opcodes in this group. The valid values are:
  - **ValidationRequired**, which specifies to validate the input and output XML. You can limit validation to just the input XML or just the output XML by using the **validationAttr** entry.
  - **ValidationNotRequired**, which specifies to skip the validation process.
  - **InternalSchemaValidation**, which specifies that the utility determines whether to validate the input and output XML by reading the validation requirement from the interaction specification.

  The default is **ValidationRequired**.

- **validationAttr** specifies whether to validate only the input XML or only the output XML. This entry is valid only when the **validation** entry is set to **ValidationRequired**. The valid values are:
  - **InputValidationNotRequired**, which specifies to validate only the output XML.
  - **OutputValidationNotRequired**, which specifies to validate only the input XML.

Defining Settings for Individual Opcodes

You use the **opcode name** lines to list the opcodes that are in the Web service. Each opcode in the Web service must have its own **opcode name** entry.

> **Note:**
>
> The opcode-specific settings in this section override the default settings in the **group-info name** entry. If you set both default values and values specific to an individual opcode, the utility uses the opcode-specific settings first.

```
 <opcode name="PCM_OP_SEARCH"
base="True"
custom="no"
opcodeNumber="89"
includePath="../schemas/"
wsdlFileInFix="Base"
validation="ValidationRequired"
validationAttr="InputValidationNotRequired">
   <opcodeFlag>1</opcodeFlag>
```

where:

- **opcode name** specifies the name of an opcode in the Web service.
- **base** specifies whether the opcode is a base opcode. See "Base Opcodes" in *BRM Developer's Guide*. The default is **False**.
- **custom** specifies whether the opcode is a custom opcode. The valid values are **yes** and **no**. The default is **no**.
- **opcodeNumber** specifies the opcode number. This field is mandatory if the **custom** field is set to **yes**.
- **includePath** specifies the relative path to the schema files. The utility adds this value to the WSDL file's **include** entries.
- **wsdlFileInFix** specifies the value to add to the WSDL file name, using the following syntax: **BRM***value***Services.wsdl**. For example, if **wsdlFileInFix** is set to **Bill**, the WSDL file name is **BRMBillServices.wsdl**. If this attribute is not present, the utility uses the **group-info name** value in the file name's syntax.
- **validation** specifies whether the adapter validates the input and output XML. The valid values are:

  - **ValidationRequired**, which specifies to validate the input and output XML. You can limit validation to just the input XML or just the output XML by using the **validationAttr** entry.
  - **ValidationNotRequired**, which specifies to skip the validation process.
  - **InternalSchemaValidation**, which specifies that the utility determines whether to validate the input and output XML by reading the validation requirements from the interaction specification.

  The default is **ValidationRequired**.

- **validationAttr** specifies whether to validate only the input XML or only the output XML. This entry is valid only when the **validation** entry is set to **ValidationRequired**. The valid values are:
  - **InputValidationNotRequired**, which specifies to validate only the output XML.
  - **OutputValidationNotRequired**, which specifies to validate only the input XML.
- **opcodeFlag** is the flag to pass to the opcode when it is called by the adapter. The default is **0**.

Sample Web Services Group Configuration File

The following shows a sample XML configuration file:

```
<?xml version='1.0'?>
<wsdl-generate>
  <namespaces>
    <partnerlink-namespace prefix="plt">http://schemas.xmlsoap.org/ws/
2003/05/partner-link/</partnerlink-namespace>
    <jca-namespace prefix="jca">http://xmlns.oracle.com/pcbpel/wsdl/jca/</
jca-namespace>
    <soap-namespace prefix="soap">http://schemas.xmlsoap.org/wsdl/soap/</
soap-namespace>
    <default-namespace>http://schemas.xmlsoap.org/wsdl/</default-namespace>
    <brm-nameSpace prefix="brm">http://xmlns.oracle.com/BRM/schemas/
BusinessOpcodes</brm-nameSpace>
  </namespaces>
  <groups>
    <group-info name="BAL" includePath="../schemas/" wsdlFileInFix="Bal"
     validation="ValidationRequired"
validationAttr="InputValidationNotRequired">
    <opcode name="PCM_OP_BAL_GET_BAL_GRP_AND_SVC"
     validation="ValidationNotRequired">
      <opcodeFlag>1</opcodeFlag>
    </group-info>
  </groups>
  <binding-info>
    <!-- binding information specific to J2CA and SOAP.
    The general binding information is filled by the tool. -->
  <JCA>
    <InteractionSpec>oracle.tip.adapter.brm.BRMInteractionSpec</
InteractionSpec>
  </JCA>
  <SOAP>
```

```
    <Action>http://localhost/operationName</Action>
    <encodingStyle>http://schemas.xmlsoap.org/soap/encoding/</encodingStyle>
  </SOAP>
  </binding-info>
  <service-info>
    <!-- Service information specific to J2CA and SOAP -->
    <JCA>
      <!-- JNDI name of BRM JCA Resource Adapter -->
      <location>eis/BRM</location>
    </JCA>
    <SOAP>
      <address>http://localhost:8080/brm/BRMWS</address>
    </SOAP>
  </service-info>
</wsdl-generate>
```

Generating WSDL Files for Web Services

After you define a Web service in the XML configuration file.

To generate WSDL files:

1. Go to the *BRM_home***/apps/brm_integrations/config** directory .
2. Run the following command:

   pin_wsdl_generator [**-c** *config_file*] [**-j** | **-s**| **-s XML**]

   where:

   - *config_file* is the name and location of the XML configuration file that describes the grouping of opcodes. By default, the utility uses the *BRM_home***/apps/brm_integrations/config/pin_wsdl_generator.xml** file.
   - **-j** generates WSDL files with JCA bindings. This is the default.
   - **-s** generates WSDL files with SOAP bindings for opcodes that take the payload as an XML string.
   - **-s XML** generates WSDL files with SOAP bindings for opcodes that take the payload as an XML element.

The utility generates the WSDL files in the directory from which the utility is run.

Configuring How to Represent Infinite Date Values

In some external applications, the infinite date value is represented as a NULL (empty XML element) value and in other external applications as the epoch time (01-01-1970 1200 AM UTC).

By default, when JCA Resource Adapter sends data to your external application, the infinite date value is the start of the epoch time.

> **Note:**
>
> Before configuring the new connection factory field, you must redeploy JCA Resource Adapter on the server running Oracle WebLogic Server.

To configure how to represent infinite date values:

1. Start the application server or the J2EE instance.
2. From the WebLogic Server Administration Console page, go to the resource adapter home page and create connection factories.
3. Do one of the following:
   - To send the date field value as NULL to represent an infinite start or end date, set **ZeroEpochAsNull** to **true**
   - To send the date field value as the start of the epoch time, set **ZeroEpochAsNull** to **false**. This is the default.

> **Note:**
>
> Connection factory field values must be lowercase.

Deploying and Configuring JCA Resource Adapter on Oracle WebLogic Server

To deploy and configure the adapter on Oracle WebLogic Server, perform the following tasks:

- Deploying JCA Resource Adapter on Oracle WebLogic Server
- Connecting JCA Resource Adapter to BRM from Oracle WebLogic Server
- Changing the JCA Resource Adapter Transaction Mode on Oracle WebLogic Server
- Configuring JCA Resource Adapter Logging on Oracle WebLogic Server

> **Note:**
>
> Any time you change the adapter's configuration, you must restart Oracle WebLogic Server for the changes to take effect.

Deploying JCA Resource Adapter on Oracle WebLogic Server

The adapter is dependent on JAR files to deploy properly. Table 3-2 lists the JAR files that the adapter requires from each application in your system.

Table 3-2 JAR File Details

Application	JAR Files
J2EE application server	**classes12.jar** **connector15.jar** **jta.jar**
Oracle BPEL process	**bpm-infra.jar** **orabpel-thirdparty.jar** **orabpel.jar** **orabpel-common.jar** **xmlparserv2.jar**
Apache	**xercesImpl.jar**

If you are deploying the adapter in a standalone WebLogic Server instance, make sure these JAR files are available to the class loader that is loading the adapter.

To deploy the adapter on WebLogic Server:

1. Start the WebLogic Server domain, if it is not already started.
2. Log in to the WebLogic Server Administration Console. The default is the following:

   ```
   http://localhost:8001/console
   ```

3. Click **Lock and Edit**.
4. In the **Domain Structure** tree, click **Deployments**.

   The Summary of Deployments pane appears.
5. Click **Install**.

   The Install Application Assistant pane appears.
6. Browse to the deploy directory of the ResourceAdapterDeployment project, select the **OracleBRMJCA15Adapter.rar** file, and click **Next**.
7. Select **Install this deployment as an application** and click **Next**.
8. In the server list, select **Server** and click **Next**.
9. Accept all other defaults on the remaining screens.
10. Click **Finish**.
11. Click **Activate Changes**, which deploys the resource adapter.

Check the state of the **OracleBRMJCA15Adapter** application in WebLogic Server Administration Console. Start the application if the state is set to **Prepared**.

Connecting JCA Resource Adapter to BRM from Oracle WebLogic Server

You connect the adapter to the BRM software by creating connection pools and connection factories. As part of the adapter deployment, the application server creates a **weblogic-ra.xml** file from the packaged **ra.xml** file.

To configure the JCA Resource Adapter connection factory and connection pools:

1. Use the resource adapter home page from WebLogic Server Administration Console to create connection pools and connection factories.

   For each connection factory, specify the following along with the needed connection factory field values in Table 3-3:

   - The JNDI location for the connection factory
   - The connection pool to use
   - How to connect to BRM by using these entries

> **Note:**
>
> Connection factory field values must be lowercase.

Table 3-3 Connection Factory Field Values

Entry	Description
ConnectionString	Specify the protocol, host name, and port number for connecting to the BRM software. For example: **ip server1 12006**. **Note:** The protocol must be set to **ip**.
InputValidation	Specifies whether to validate the input XMLRecord: • **true** : The adapter validates the input XMLRecord against the opcode schema. • **false** : The adapter does not validate the input XMLRecord. The default is **false**. This overrides any other validation parameter specified in the WSDL file. For information about setting the validation parameters in the WSDL file, see "Defining a Web Service".
OutputValidation	Specifies whether to validate the output XMLRecord: • **true**: The adapter validates the output XMLRecord against the opcode schema. • **false**: The adapter does not validate the output XMLRecord. The default is **false**. This overrides any other validation parameter specified in the WSDL file. For information about setting the validation parameters in the WSDL file, see "Defining a Web Service".
LoginType	Specifies the authentication method: • **1**: The adapter logs in to BRM by using the specified login name and password. • **0**: The adapter logs in to BRM by using the specified service type and POID ID. The default is **1**.

Table 3-3 (Cont.) Connection Factory Field Values

Entry	Description
UserName	Specifies the login name the adapter uses for logging in to the BRM software. **Note:** This entry is required only if **LoginType** is set to **1**.
Password	Specify the password the adapter uses for logging in to the BRM software. **Note:** This entry is required only if **LoginType** is set to **1**.
PoidID	Specifies the POID. This entry should be set to **1**.
ServiceType	Specifies the service the adapter uses to log in to the BRM software. The default is **/service/pcm_client**.
BRMConnectionPoolMaxSize	Specifies the maximum number of connections the connection pool can create.
BRMConnectionPoolMinSize	Specifies the minimum number of connections created by the connection pool when the pool is initialized.
transactionMode	Specifies the transaction mode used for message producers. See "About BRM JCA Resource Adapter Transaction Management".
MultiDB	Specifies whether to enable connections to multiple database schemas: • **true**: Enables connections to multiple schemas. • **false**: Disables connections to multiple schemas. When the **MultiDB** connection factory entry is set to **true**, you do not need to supply the target schema's database number; the adapter automatically opens transactions on the correct schema and generates an error if a transaction attempts to manipulate data across schemas.
BRMConnectionPoolTimeout	Specifies the maximum amount of time in milliseconds that a connection request is queued.
BRMConnectionPoolMaxIdleTime	Specifies the maximum amount of time in milliseconds that a free connection in the connection pool can be idle. If a free connection is idle for a time greater than or equal to **BRMConnectionPoolMaxIdleTime**, the connection is removed from the pool. **Note:** Only the connections created after the limit set by **BRMConnectionPoolMinsize** is reached are verified for the idle time; if the condition is met, the connection is removed from the pool.

Table 3-3 (Cont.) Connection Factory Field Values

Entry	Description
InteractionTimeZone	Specifies the time zone of the date fields in which JCA Resource Adapter sends the date fields to the Oracle AIA components. • **TZ names:** Time zone set on the BRM server, if it is different than the JCA server time. For example, EST, PST, or CST. • **UTC:** Coordinated Universal Time is the time standard used across the world. • **LOCAL:** Specified to use the time zone of the server on which JCA Resource Adapter is running. JCA Resource Adapter converts date fields to the time zone set by **InteractionTimeZone** on outbound messages.
MaxRequestListSize	Specifies the maximum number of connection requests the connection pool can queue before returning a NAP_CONNECT_FAILED error. Typically, the requests are queued when all the connections in the connection pool are occupied.
AverageOpcodeCount	Specifies the average number of opcode calls per thread.
zeroEpochAsNull	Specifies how JCA Resource Adapter sets infinite date values. The default is **false**.

2. Save and close the file.

You have successfully configured the adapter to connect to BRM.

Changing the JCA Resource Adapter Transaction Mode on Oracle WebLogic Server

By default, JCA Resource Adapter is deployed in XA Transaction mode. After the adapter is deployed, you can change the transaction mode.

For information about the transaction modes, see "About BRM JCA Resource Adapter Transaction Management".

To change the JCA Resource Adapter transaction mode on the WebLogic server:

1. Start the WebLogic Server domain if it is not already started.
2. Log in to WebLogic Server Administration Console. The default is the following:

   ```
   http://localhost:8001/console
   ```

3. In the **Domain Structure** tree, click **Deployments**.

   The Summary of Deployments pane appears.
4. In the Deployments table, click the JCA Resource Adapter name.

   By default, the name is **OracleBRMJCA15Adapter**.

The Settings for *Adapter_Name* pane appears.

5. Click the **Configuration** tab.
6. Click the **Outbound Connection Pool** tab.
7. In the Outbound Connection Pool Configuration table, expand the **OracleConnectionFactory** node.
8. Click **eis/BRM**, which is the name of the JNDI for JCA Resource Adapter.
9. In the **Properties** tab, click the **TransactionMode** property value.
10. Enter one of the following values:
    - For No Transaction mode, enter **NO_TRANSACTION**.

      In this mode, the transaction is managed by the application, such as Oracle AIA, that is using JCA Resource Adapter to connect to BRM.
    - For Local Transaction mode, enter **LOCAL_TRANSACTION**.

      In this mode, the adapter supports only one-phase commit local transactions managed by the application server, such as Oracle WebLogic Server.
    - For XA Transaction mode, enter **XA_TRANSACTION**.

      (Default) In this mode, the adapter supports both two-phase commit XA transactions and one-phase commit transactions managed by a global transaction manager through the XAResource interface. See "About XA Transaction Support".
11. Click **Save**.
12. Click the **Transaction** tab.
13. From the **Transaction Support** list, select the value that matches the transaction mode selected in step 10:
    - **No Transaction** for No Transaction mode
    - **Local Transaction** for Local Transaction mode
    - **XA Transaction** for XA Transaction mode
14. Click **Save**.
15. Log out of WebLogic Server Administration Console.

Configuring JCA Resource Adapter Logging on Oracle WebLogic Server

JCA Resource Adapter supports Java Unified Logging (JUL), which enables the adapter to use the Java Unified Logging library.

Creating a Startup Class

To create a Startup class:

1. Copy the **weblogic_startup.jar** file provided with the installation package to *BRM_home***/apps/brm_integrations/jca_adapter**.

2. Log in to WebLogic Server Administration Console. The default is the following:

   ```
   http://localhost:8001/console
   ```

3. Click **Lock and Edit**.
4. In the **Domain Structure** tree, expand **Environment** and then click **Startup and Shutdown classes**.

   The Startup and Shutdown Classes pane appears.
5. Click **New**.

   The Configure a New Startup or Shutdown Class: Class Type pane appears.
6. Select **Startup Class** and click **Next**.

   The Configure a New Startup or Shutdown Class: Startup Class Properties pane appears.
7. In the **Name** field, enter **BRMStartupClass**.
8. In the **Class Name** field, enter **oracle.tip.adapter.brm.BRMLoggerStartUP**.
9. Click **Next**.

   The Configure a New Startup or Shutdown Class: Select Targets pane appears.
10. From the **Servers** box, select the server on which to deploy the class and click **Finish**.

    The Startup and Shutdown Classes pane appears.
11. Click **BRMStartupClass**.

    The Settings for BRMStartupClass pane appears.
12. Select **Run Before Application Activations** and click **Save**.
13. Click **Activate Changes**.
14. Restart WebLogic Server to apply the changes.

By default, log files are created in the *domain_home***/BRMAdapterLogs** directory, where *domain_home* is the directory in which the WebLogic Server domain is configured.

Changing the Java Logging Level on Oracle WebLogic Server

You change the logging level by using JConsole.

1. Go to the *WebLogic_Home***/jdk160_05/bin** directory, where *WebLogic_Home* is the directory in which you installed the WebLogic Server.
2. Enter the following command:

   ```
   jconsole
   ```

   The New Connection dialog box appears.
3. Select **Remote Process**, enter the WebLogic Server host name and port number, enter your user name and password, and then click **Connect**.

> **Note:**
>
> When WebLogic Server is running on the same system, you can use **Local Process** without authentication. The **Local Process** list shows the WebLogic Server process name and PID.

The Java Monitoring and Management Console pane appears.

4. Click the **MBeans** tab.
5. In the **MBean** tree, expand **java.util.logging**, then expand **Attributes**, and then select **LoggerNames**.
6. Copy the **oracle.tip.adapter.brm.BRMConnectionFactory** line.
7. In the **MBean** tree, expand **java.util.logging** and then select **Operations**.

   The Operation invocation pane appears.
8. In the **void setLoggerLevel p0** and **p1** fields, paste the logger name and log level that you copied in step 6.
9. Click **setLoggerLevel**, which updates the log level.
10. Close the Java Monitoring and Management Console pane.

For more information, see "Using JConsole" in *Java SE Monitoring and Management Guide*.

Changing the XA Transaction Timeout Period

By default, successfully prepared XA transactions expire in BRM if a commit request is not received within a specified time after the transaction opens. The XA transaction timeout is specified in the following places, which are listed in the order of priority:

- Application server configuration settings. For information about changing this timeout period, see your application server documentation.
- Oracle DM configuration file **dm_xa_trans_timeout_in_secs** entry. This timeout is used only if the application server XA transaction timeout is not specified.

  To change the Oracle DM XA transaction timeout period:

  1. Open the *BRM_home*/**sys/dm_oracle/pin.conf** file in a text editor.
  2. If necessary, add the **dm_xa_trans_timeout_in_secs** entry to the file:

     ```
     - dm dm_xa_trans_timeout_in_secs seconds
     ```

     where *seconds* is the number of seconds after the transaction is opened that it expires. The minimum value is **10**, and the maximum value is **5184000** (60 days). The default value is **3600** (1 hour).
  3. Save and close the file.
  4. Restart the Oracle DM instance.

Testing JCA Resource Adapter Configuration and BRM Connectivity

Use the **BRMAdapterServletClient** application to test the JCA Resource Adapter configuration and connectivity to BRM. The application enables you to provide input to an opcode and view its output.

To set up the **BRMAdapterServletClient** application, deploy and configure it on your Oracle WebLogic Server.

> **Note:**
>
> Make sure JCA Resource Adapter is deployed. Set the deployment order of the **BRMAdapterServletClient** application higher than the deployment order of JCA Resource Adapter.

To deploy **BRMAdapterServletClient** application on your application server:

1. Copy the **BRMAdapterServletClient.war** file from the *BRM_home***/apps/brm_integrations/jca_adapter** directory to a location that is accessible to the adapter deployment tool.
2. Specify to deploy the **BRMAdapterServletClient** application in the same instance as that of JCA Resource Adapter.
3. Upload the **BRMAdapterServletClient.war** archive file.
4. Run and test the adapter. You can access the application's default page at the following URL:

   ```
   http://HostName:PortNumber/BRMAdapterServletClient
   ```

   where *HostName* is the host name of the machine where the application is deployed and *PortNumber* is the port where the application is deployed.

After the application is deployed, you can begin using it to run and test the adapter. You can access the application's default page at the following URL:

```
http://host_name:port_number/BRMAdapterServletClient
```

where *host_name* is the host name of the machine where the application is deployed and *port_number* is the port where the application is deployed.

Manually Committing or Rolling Back XA Transactions

If an XA transaction expires after being successfully prepared, a system administrator with the SYSDBA privilege can manually commit or roll back the transaction.

To commit or roll back XA transactions manually:

1. Using SQL*Plus, log in to your database as the SYSDBA user:

   ```
   sqlplus sys@databaseAlias as sysdba
   ```

   where *databaseAlias* is the database alias of your BRM database.

2. Enter the following command, which retrieves the local transaction IDs for all pending XA transactions that were successfully prepared:

   ```
   select LOCAL_TRAN_ID,  GLOBAL_TRAN_ID from DBA_2PC_PENDING where
   STATE = 'prepared';
   ```

   Timed-out XA transactions are stored in the DBA_2PC_PENDING view. See "Monitoring Global Transactions" in *BRM System Administrator's Guide*.

3. Enter one of the following commands:

   - To commit the transaction:

     ```
     commit force 'LocalTranID';
     ```

   - To roll back the transaction:

     ```
     rollback force 'LocalTranID';
     ```

   where *LocalTranID* is the ID of the local transaction branch that you are committing or rolling back.

   > **Note:**
   >
   > JCA Resource Adapter uses the global transaction ID to identify the global transaction to which the local transaction branch belongs.

4. If your global transaction manager does not issue forget requests, purge the manually processed transaction from the DBA_2PC_PENDING view by using the EXECUTE DBMS_TRANSACTION.PURGE_LOST_DB_ENTRY() procedure.

   For more information, see the discussion about how to purge a distributed transaction from a database (Doc ID 159377.1) on the My Oracle Support Web site:

   https://support.oracle.com

4

JCA Resource Adapter Utilities

This document provides reference information for Oracle Communications Billing and Revenue Management (BRM) JCA Resource Adapter and Web Services Manager utilities.

Topics in this document:

pin_dd_to_schema

Use the **pin_dd_to_schema** utility to generate the XSD schema for your storable classes and subclasses.

For more information, see "Generating the Schema for Your Storable Classes and Subclasses".

> **Note:**
>
> To connect to the BRM database, the **pin_dd_to_schema** utility needs a configuration file in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
pin_dd_to_schema  -e config_file | -r storable_class | -h
```

Parameters

-e *config_file*
Generates the schema for all storable classes listed in the configuration file and their subclasses. For example, if the configuration file lists the **/service/telco/gsm** storable class, the utility generates the schema for **/service/telco/gsm**, **/service/telco/gsm/telephony**, **/service/telco/gsm/sms**, and any other **/service/telco/gsm** subclasses in your system.

> **Note:**
>
> You must specify both the name and path of the configuration file.

-r *storable_class*
Generates the schema for the specified storable class and all of its subclasses. For example:

```
pin_dd_to_schema -r /service/telco/gsm
```

-h
Displays the syntax and parameters for this utility.

Results

The **pin_dd_to_schema** utility generates the output files in the directory from which it was run.

The utility notifies you only if it encounters errors. For errors, look in the **default.pinlog** file, which is created in the directory from which the utility was run.

> **Note:**
>
> If you receive the following error when running **pin_dd_to_schema** on Linux systems, set the LC_ALL environment variable to C (**setenv LC_ALL C**) and then rerun the utility.
>
> ```
> Malformed UTF-8 character
> (unexpected continuation byte 0xac, with no preceding start
> byte)
> in bitwise and (&) at BRM_home/lib/5.8.0/Switch.pm line 251.
> Malformed UTF-8 character (unexpected continuation byte 0xab,
> with
> no preceding start byte) in bitwise and (&).
> ```

pin_opspec_to_schema

Use the **pin_opspec_to_schema** utility to generate XSD schema files for opcodes.

For more information, see "Generating the Schema for Your Opcodes".

Location

BRM_home/**bin**

Syntax

pin_opspec_to_schema -i *input_file* [**-o** *output_file*] [**-h**]

Parameters

-i *input_file*
Specifies the name and location of the opcode XML flist specification file to convert into XSD schema. If you do not specify the absolute path to the file, the utility looks in the current directory.

-o *output_file*
Creates the XSD schema output file using the name and location you specify. By default, the utility generates an output file named *opcode_name***.xsd** in the directory from which you run the utility.

-h
Displays the syntax and parameters for this utility.

Results

The **pin_opspec_to_schema** utility notifies you when it successfully generates schema files. Any errors are displayed on the console.

pin_opspec_to_schema_v2

Use the **pin_opspec_to_schema_v2** utility to generate XSD schema files for opcodes that take payload as XML element.

For more information, see "Generating the Schema for Your Opcodes".

Location

*BRM_home***/bin**

Syntax

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

Parameters

input_file
Specifies the name and location of the opcode XML flist specification file to convert into XSD schema. If you do not specify the absolute path to the file, the utility looks in the current directory.

output_file
Creates the XSD schema output file using the name and location you specify. By default, the utility generates an output file named *opcode_name***.xsd** in the directory from which you run the utility.

Results

The **pin_opspec_to_schema_v2** utility notifies you when it successfully generates schema files. Any errors are displayed on the console.

pin_wsdl_generator

Use the **pin_wsdl_generator** utility to generate WSDL files for Web services.

For more information, see "Generating WSDL Files for Web Services".

Location

BRM_home/**bin**

Syntax

```
pin_wsdl_generator [-c config_file] [-j | -s] [-s XML] [-d] [-h]
```

Parameters

-c *config_file*
Specifies the name and location of the XML configuration file that describes how to group opcodes into Web services. By default, the utility uses the *BRM_home*/**apps/brm_integrations/config/pin_wsdl_generator.xml** file. See "Defining a Web Service".

-j | -s
Specifies whether to create WSDL files with JCA (**-j**) or SOAP (**-s**) bindings. The default is to generate WSDL files with JCA bindings.

-s XML
Creates WSDL files for Web services that take payload as XML element. For example:

```
pin_wsdl_generator -c pin_wsdl_generator.xml -s XML
```

-d
Runs in debug mode and displays more detailed messages.

-h
Displays the syntax and parameters for this utility.

Results

The **pin_wsdl_generator** utility generates the output WSDL files in the directory from which it was run.

The utility notifies you when it successfully generates WSDL files. Any errors are displayed on the console.