Oracle® Application Server Integration InterConnect

Adapter for WebSphere MQ Installation and User's Guide

10*g* Release 2 (10.1.2)

B14072-02

December 2005

Oracle Application Server Integration InterConnect Adapter for WebSphere MQ Installation and User’s Guide, 10*g* Release 2 (10.1.2)

B14072-02

Primary Author: Rima Dave

Contributing Author: Vimmy Raj

Contributor: Ashwin Patel, Maneesh Joshi, Rahul Pathak, Harish Sriramulu

Contents

Preface

This Preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

Oracle Application Server Integration InterConnect Adapter for WebSphere MQ Installation and User's Guide is intended for those who perform the following tasks:

- install applications
- maintain applications

To use this document, you need to know how to install and configure OracleAS Integration InterConnect.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, refer to these Oracle resources:

- *Oracle Application Server Integration InterConnect User's Guide*
- *Oracle Application Server Integration InterConnect Installation Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction

This chapter provides an overview of the Oracle Application Server Integration InterConnect (OracleAS Integration InterConnect) Adapter for IBM WebSphere MQ (WebSphere MQ adapter). It contains the following topics:

- WebSphere MQ Adapter Overview
- WebSphere MQ Adapter System Requirements
- WebSphere MQ Adapter Knowledge Requirements
- WebSphere MQ Adapter Interfaces
- Known WebSphere MQ Adapter Limitations

1.1 WebSphere MQ Adapter Overview

The WebSphere MQ adapter enables OracleAS Integration InterConnect to send and receive messages from the WebSphere MQ queues and topics. This enables an application that uses IBM's WebSphere MQ as its messaging technology to be integrated with other applications using OracleAS Integration InterConnect. As a result, the WebSphere MQ adapter is useful in all enterprise application integration scenarios involving WebSphere MQ-based applications.

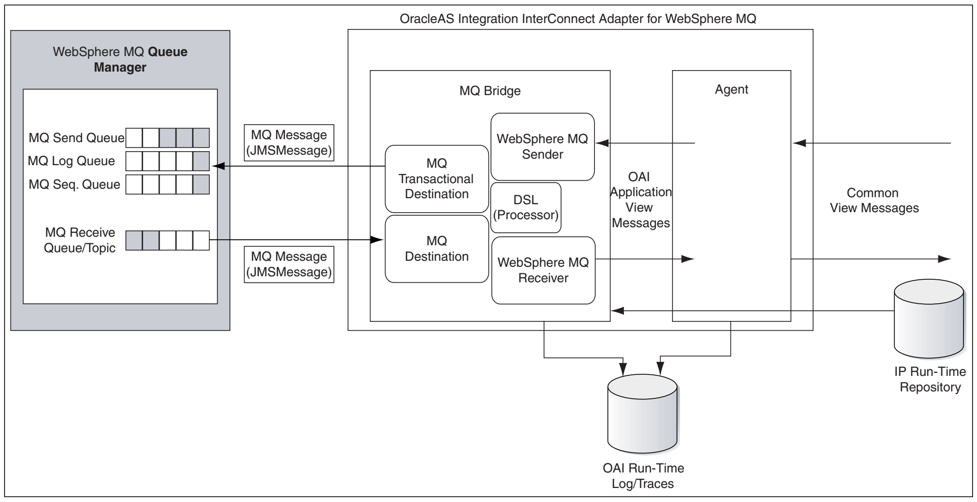
The WebSphere MQ adapter is primarily used to provide messaging capabilities between OracleAS Integration InterConnect and the WebSphere MQ queuing systems. These capabilities include support for the publish and subscribe paradigm for message exchange, such as, sending or receiving orders, invoices, and product updates.

In addition to the basic publish and subscribe messaging, the WebSphere MQ adapter also supports the OracleAS Integration InterConnect request and reply paradigm, which maps directly to WebSphere MQ's own generic support for request and reply messages. This capability is based on the support for message correlation for OracleAS Integration InterConnect as well as in WebSphere MQ. Examples include inventory lookups, price calculations, and status requests.

This guide explains all necessary design-time and run-time concepts of the WebSphere MQ adapter.

Figure 1–1 describes how the WebSphere MQ adapter interacts with an WebSphere MQ queue manager and OracleAS Integration InterConnect.

Figure 1–1 How the WebSphere MQ Adapter Interacts with an WebSphere MQ Queue Manager and OracleAS Integration InterConnect



The WebSphere MQ adapter uses four logical queues or destinations to support its interaction with the WebSphere MQ messaging system: three for inbound messages, from OracleAS Integration InterConnect to WebSphere MQ, and one for outbound messages, from WebSphere MQ to OracleAS Integration InterConnect.

Three queues are required for inbound messages to support sending of messages from OracleAS Integration InterConnect to WebSphere MQ in a transactionally safe manner. The queues are used in the following manner:

- One queue is the actual destination for inbound messages.
- One queue is used to keep a log of already received messages within a transaction.
- One queue is used to hold and generate incrementally unique transaction IDs.

The two logical queues, for the logs and transaction IDs, can be combined into one physical queue.

See Also: "Inbound" on page 1-4

1.2 WebSphere MQ Adapter System Requirements

The following sections describe WebSphere MQ adapter system requirements:

- Hardware Requirements
- Software Requirements

1.2.1 Hardware Requirements

Table 1–1 lists the hardware requirements for installing the Oracle WebSphere MQ adapter.

Table 1–1 Hardware Requirements

Hardware	Windows	UNIX
Disk Space	400 MB	400 MB
Memory	512 MB	512 MB

1.2.2 Software Requirements

The following sections list software requirements for the WebSphere MQ adapter:

- Operating System Requirements
- JRE Requirements
- WebSphere MQ adapter Requirements

Operating System Requirements

Table 1–2 lists operating system requirements for installing the WebSphere MQ adapter.

Table 1–2 Operating System Requirements

Operating System	Version
HP Tru64	HP Tru64 UNIX (Alpha) 5.1b
HP-UX	HP-UX (PA-RISC) 11.11, 11.23
IBM AIX	AIX (POWER) version 5.2
Linux (x86)	Red Hat Enterprise Linux 2.1, 3.0 SuSE SLES8, SLES9
Sun SPARC Solaris	Sun SPARC Solaris 2.8 and 2.9
Microsoft Windows	Windows XP Professional, Windows 2000(SP3 or higher)

JRE Requirements

OracleAS Integration InterConnect uses Java Runtime Environment (JRE) 1.4, which is installed with its components.

WebSphere MQ adapter Requirements

Table 1–3 lists the minimum software requirements for installing the WebSphere MQ adapter.

Table 1–3 WebSphere MQ Requirements

Software	Supported Versions
WebSphere MQ: Publish/Subscribe	Version 1.1 or latest supporting WebSphere MQ 5.2 or 5.3
WebSphere MQ classes for Java and WebSphere MQ classes for Java Message Service (JMS)	Only required with WebSphere MQ Version 5.2.x. WebSphere MQ Version 5.3 Client includes Java/JMS.
WebSphere MQ Client	Version 5.2 or 5.3
WebSphere MQ Server	Version 5.2 or 5.3

1.3 WebSphere MQ Adapter Knowledge Requirements

The installation of the WebSphere MQ adapter software components mentioned in WebSphere MQ adapter Requirements on page 1-3 should be performed by a WebSphere MQ system administrator.

To configure and use the WebSphere MQ adapter, you require the following:

- Basic WebSphere MQ administration skills, such as starting the listener and creating queues.
- Basic knowledge of WebSphere MQ connectivity concepts, like channel and client.
- Basic knowledge of WebSphere MQ Java and JMS, for example, WebSphere MQ JMS queue and topic URI syntax.

You must create and start the WebSphere MQ queues and topics referred to in this guide before starting the WebSphere MQ adapter.

1.4 WebSphere MQ Adapter Interfaces

The following sections describe the WebSphere MQ adapter interfaces.

- General
- Inbound
- Outbound
- Connection Types

1.4.1 General

The WebSphere MQ adapter uses the WebSphere MQ JMS URI syntax for specifying the queues and topics that constitute the endpoints for inbound and outbound messages.

This format is derived from Uniform Resource Identifiers (URIs) and enables you to specify remote queues and set other queue connection properties. Remote queues are on a queue manager other than the one to which you have connected.

The syntax for the queue URI is as follows:

```
queue://[queue-manager]/queue[?property=value [&property=value ]*]
```

The URI for a queue starts with the sequence `queue://`, followed by the name of the queue manager where the queue resides, a further `/`, followed by the name of the queue, and optionally, a list of name-value pairs to set the remaining queue properties.

If the name of the queue manager is omitted, then the default queue manager, as specified in the `adapter.ini` file, is used.

The syntax for the topic URI is as follows:

```
topic://SAP/Events/HR/newCustomer?priority=1
```

The URI for a topic starts with the sequence `topic://`, followed by the full path to the topic, and optionally, a list of name-value pairs to set the remaining queue properties. The topic URI syntax does not specify the queue manager. It must be specified in the `mq.default.queue_manager` property in the `adapter.ini` file.

1.4.2 Inbound

Inbound interfaces consist of WebSphere MQ queues to which messages are sent by the WebSphere MQ adapter. The WebSphere MQ adapter only supports WebSphere MQ queues, not topics, for *inbound* interfaces, because of the following constraints:

- The WebSphere MQ adapter must send messages to WebSphere MQ in a transactionally safe manner, because it implements the OracleAS Integration

InterConnect SDK TransactionalMessageReceiver interface. This requires the use of a queue for keeping log records

- The destination queue or topic and log queue must be updated within the same JMS transaction.
- The WebSphere MQ JMS implementation does not support Universal JMS sessions, which would allow queues and topics to be updated within the same transaction.
- Storing temporary log records in a topic is not practical.

1.4.3 Outbound

Outbound interfaces can consist of both queues and topics from which the WebSphere MQ adapter will receive messages. Additional configuration parameters in the `adapter.ini` file allow for defining a JMS selector expression, which can be used to filter messages that should be received by WebSphere MQ adapter. Another parameter controls whether the message consumption should be performed within a transactional or nontransactional JMS session.

1.4.4 Connection Types

WebSphere MQ supports the following connection types:

- Local (bind)
- Remote (client)

1.4.4.1 Local Connections

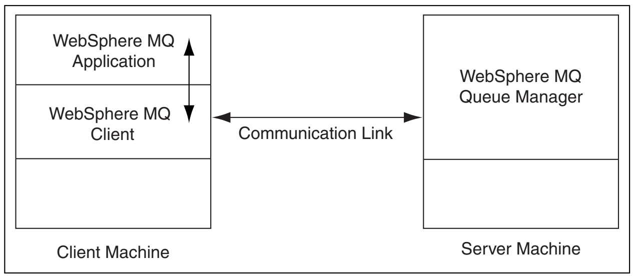
For local connections, the WebSphere MQ queue manager runs on the same host as the WebSphere MQ adapter. In this case, the WebSphere MQ adapter only needs to know the queue name and the queue manager name in order to establish a queue connection.

1.4.4.2 Remote Connections

For remote connections, the WebSphere MQ queue manager runs on a different host. In this case, the WebSphere MQ adapter needs WebSphere MQ client libraries, which must be installed separately, in order to establish a queue connection. The WebSphere MQ adapter also needs additional configuration information, such as the name of the remote host, the port number where the WebSphere MQ listener is listening, and the channel name.

Figure 1–2 displays a client connection.

Figure 1–2 Client Connection



1.5 Known WebSphere MQ Adapter Limitations

The WebSphere MQ adapter has the following limitations:

- Encryption is not supported.
- All message types other than ObjectMessage JMS are supported.
- WebSphere MQ message grouping and segmentation are not supported.
- JMS Message properties of received messages from WebSphere MQ are not passed on to OracleAS Integration InterConnect. They can be useful in selecting a relevant D3L transformation.
- WebSphere MQ transactions are used to support the OracleAS Integration InterConnect `TransactionalMessageReceiver` interface. The `mq.default.trans_id_expiry` configuration parameter determines how long a transaction started by the OracleAS Integration InterConnect Agent can stay idle before it expires. WebSphere MQ does not expose the concept of a persistent transaction identifier, as a result, the transaction identifier is only valid for the lifespan of the WebSphere MQ adapter instance and the underlying transactional JMS session. Consequently, a given transaction ID is rendered invalid immediately when the adapter process dies.
- An WebSphere MQ adapter instance only supports one outgoing (sending) endpoint. For example, it can only communicate with one queue manager.

2 Installation and Configuration

This chapter describes how to install and configure the WebSphere MQ adapter. It contains the following topics:

- Installing the WebSphere MQ Adapter
- Installing Multiple WebSphere MQ Adapters in the Same Oracle Home
- Configuring the WebSphere MQ Adapter
- Uninstalling the WebSphere MQ Adapter

2.1 Installing the WebSphere MQ Adapter

The WebSphere MQ adapter must be installed in an existing Oracle home Middle Tier for OracleAS Integration InterConnect 10*g* Release 2 (10.1.2).

This section describes the following topics:

- Preinstallation Tasks
- Installation Tasks

2.1.1 Preinstallation Tasks

Before installation, ensure that the WebSphere MQ server is installed. If the WebSphere MQ server is running on a remote host, then ensure that the WebSphere MQ client is installed.

The IBM WebSphere MQ installation guides for various platforms are available at:

`http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp`

The README files for various plaforms are available at:

`http://www-306.ibm.com/software/integration/mqfamily/support/readme/`

Note: On Unix, the WebSphere MQ software should always be installed as `user mqm`, whose primary group should be `mqm`.

Verify that the WebSphere MQ system is functional before commencing the installation of the WebSphere MQ adapter.

Refer to the following guides before installing the WebSphere MQ adapter:

- *Oracle Application Server Installation Guide* for information about Oracle Universal Installer startup.

- *Oracle Application Server InterConnect Installation Guide* for information on software, hardware, and system requirements for OracleAS Integration InterConnect.

> **Note:** OracleAS Integration InterConnect Hub is installable through the OracleAS Integration InterConnect Hub installation type. You must install the OracleAS Integration InterConnect Hub before proceeding with the WebSphere MQ adapter installation.

2.1.2 Installation Tasks

To install the WebSphere MQ adapter:

1. In the Available Product Components screen of the OracleAS Integration InterConnect installation, select **OracleAs Integration InterConnect Adapter for IBM WebSphere MQ 10.1.2.0.2**, and click **Next**. The Set Oracle Wallet Password screen is displayed.
2. Enter and confirm the password, which will be used to manage OracleAS Integration InterConnect installation. Click **Next**.
   - Go to step 3, if installing the WebSphere MQ adapter in an OracleAS Middle Tier Oracle home that does not have an InterConnect component already installed. Ensure that the OracleAS Integration InterConnect hub has been installed.
   - Go to step 4, if installing the WebSphere MQ adapter in an OracleAS Middle Tier Oracle home that has an existing InterConnect component. Ensure that it is a home directory to an OracleAS Integration InterConnect component.
3. The Specify Hub Database Connection screen is displayed. Enter information in the following fields:
   - Host Name: The host name of the computer where the hub database is installed.
   - Port Number: The TNS listener port for the hub database.
   - Database SID: The SID for the hub database.
   - Password: The password for the hub database user.
4. Click **Next**. The Specify IBM WebSphere MQ Adapter Name screen is displayed.
5. Enter the application to be defined. Blank spaces are not permitted. The default value is `myMQApp`.

   > **Note:** You can change the application name in iStudio after installation. In such a case, you need to specify the password corresponding to new application name in the Oracle Wallet.
   >
   > For more informtion, refer to the following sections in Appendix A, "Frequently Asked Questions":
   >
   > - Section A.3, "My WebSphere MQ adapter is not starting. What could be the reason?"
   > - Section A.6, "How do I secure my passwords?"

6. Click **Next**. The Specify IBM WebSphere MQ Adapter Usage screen is displayed.
7. Select one of the options and go to the step specified.

If You Select...	Then Click Next and Go to Step...
Configure for both sending and receiving messages	8
Configure for sending messages ONLY	8
Configure for receiving messages ONLY	10

Note: You can change the values for these selections later by editing the parameter settings in the `adapter.ini` file.

8. Enter the following information in the Configure Sending Endpoint Information window:

   - WebSphere MQ inbound queue: The URI of the WebSphere MQ queue to which messages are sent.
   - WebSphere MQ inbound log queue: The URI of the WebSphere MQ queue that temporarily stores log records during sending transactions.
   - WebSphere MQ inbound id queue: The URI of the WebSphere MQ queue that is used to store and generate unique (sequential) transaction identifiers for the inbound or sending transactions.

   Note: The preceding URIs can only denote queues, not topics.

   The logical WebSphere MQ inbound log queue and the logical WebSphere MQ inbound id queue can refer to the same physical WebSphere MQ queue.

9. Click **Next**. The installation window that is displayed is based on the selection made in Step 7.

If You Selected...	Then Go to Step...
Configure for both sending and receiving messages	10
Configure for sending messages ONLY	11

10. Enter the WebSphere MQ outbound queue/topic information in the Configure Receiving Endpoint Information window. This is a URI for the WebSphere MQ queue or topic from which messages are received. It is used to listen to incoming messages from WebSphere MQ or as JMS ReplyTo addresses while sending request messages to WebSphere MQ.

11. Enter the following information on the Define WebSphere MQ Connection Information screen:

   - WebSphere MQ Java installation Path: This path specifies the root directory of the WebSphere MQ Java (client) installation, which typically is `/opt/mqm/java`. During startup, the WebSphere MQ Adapter will try to locate the JAR files (`jms.jar`, `com.ibm.mqjms.jar`, `com.ibm.mq.jar`, and `connector.jar`) in the `lib` subdirectory in this path..
   - WebSphere MQ Queue Manager: The name of the WebSphere MQ queue manager to which to connect.

- WebSphere MQ Client Connection Type: From the list, select the type of connection to make to the WebSphere MQ queue manager. Select Remote to use a client connection (through an WebSphere MQ channel), or select Local to bind to a queue manager running on the same computer as the adapter.

12. Click **Next**. The installation screen that is displayed is based on the selection you made in Step 11.

If You Selected...	Then Go to Step...
Remote	13
Local	14

13. Enter the following information on the Specify WebSphere MQ Client Connect Parameters screen:

    - Host Name: The DNS name of the host where the WebSphere MQ queue manager resides.
    - Port Number: The port number to connect to on the MQ Server host. The default port number is 1414. This port is defined when starting the WebSphere MQ listener by the command `runmqlsr` (for example, `runmqlsr -m qmqr -t tcp -p 1415`).
    - WebSphere MQ Channel Name: The name of the WebSphere MQ channel to use for the client connection.

14. Click **Next**. The Summary screen is displayed.
15. Click **Install** to install the WebSphere MQ adapter. The adapter is installed in the following directory:

Platform	Directory
UNIX	*ORACLE_HOME*/integration/interconnect/adapters/*Application*
Windows	*ORACLE_HOME*\integration\interconnect\adapters\Application

    You defined the value of `Application` in Step 4.

16. Click **Exit** on the Installation screen to exit the WebSphere MQ adapter installation.

2.2 Installing Multiple WebSphere MQ Adapters in the Same Oracle Home

To install multiple instances of the WebSphere MQ adapter in same Oracle home, use the `copyAdapter` script located in the *ORACLE_HOME*/integration/interconnect/bin directory.

Usage: `copyAdapter` *app1 app2*

For example, you have one instance of WebSphere MQ adapter with name `myMQApp` installed on a computer. To install another instance of the WebSphere MQ adapter with name `myMQApp1` in the same Oracle home, use the following command:

```
copyAdapter myMQApp myMQApp1
```

The `copyAdapter` script is copied to the following `bin` directory only during Hub installation:

- UNIX: `ORACLE_HOME/integration/interconnect/bin`
- Windows: `ORACLE_HOME\integration\interconnect\bin`

If you need to use this script to create multiple adapters on a spoke computer, then copy the script to the `bin` directory on the spoke computer, and edit the script to reflect the new Oracle home.

After running the `copyAdapter` script, If you want to manage or monitor the newly installed adapter through Oracle Enterprise Manager 10*g* Application Server Control Console, then you need to modify the `opmn.xml` file by adding information about the new instance. For example, you have created a new instance of the WebSphere MQ adapter `myMQApp1` by using the `copyAdapter` script. To manage the `myMQApp1` adapter through Enterprise Manager, perform the following:

1. Navigate to the *`MiddleTier`*`\bin` directory and run the following command to stop the Enterprise Manager:

   ```
   emctl stop iasconsole
   ```

2. Next, specify the information about this new instance in the `opmn.xml` file located in the *`ORACLEMIDDLETIER_HOME`*`/opmn/conf` directory as follows:

   ```
   <process-type id="myMQApp1" module-id="adapter" working-dir="$ORACLE_
   HOME/integration/interconnect/adapters/myMQApp1" status="enabled">
          <start timeout="600" retry="2"/>
          <stop timeout="120"/>
          <port id="icadapter_dmsport_range" range="15701-15800"/>
          <process-set id="myMQApp1" restart-on-death="true" numprocs="1">
               <module-data>
                   <category id="start-parameters">
                       <data id="java-parameters" value="-Xms8M"/>
                       <data id="class-name"
                        value="oracle.oai.agent.service.AgentService"/>
                   </category>
                   <category id="stop-parameters">
                       <data id="java-parameters" value="-mx64m"/>
                       <data id="class-name"
                        value="oracle.oai.agent.proxy.ShutdownAgent"/>
                       <data id="application-parameters"
                        value="persistence/Agent.ior"/>
                   </category>
               </module-data>
          </process-set>
   </process-type>
   ```

   The `opmn.xml` file would appear like this:

   ```
   <process-type id="myMQApp" module-id="adapter" working-dir="$ORACLE
   _HOME/integration/interconnect/adapters/myMQApp" status="enabled">
          <start timeout="600" retry="2"/>
          <stop timeout="120"/>
          <port id="icadapter_dmsport_range" range="15701-15800"/>
          <process-set id="myMQApp" restart-on-death="true" numprocs="1">
               <module-data>
                   <category id="start-parameters">
                       <data id="java-parameters" value="-Xms8M"/>
                       <data id="class-name"
                        value="oracle.oai.agent.service.AgentService"/>
                   </category>
                   <category id="stop-parameters">
                       <data id="java-parameters" value="-mx64m"/>
   ```

```
                    <data id="class-name"
             value="oracle.oai.agent.proxy.ShutdownAgent"/>
             <data id="application-parameters"
              value="persistence/Agent.ior"/>
         </category>
     </module-data>
  </process-set>
</process-type>

<process-type id="myMQApp1" module-id="adapter" working-dir="$ORACLE
_HOME/integration/interconnect/adapters/myMQApp1" status="enabled">
       <start timeout="600" retry="2"/>
       <stop timeout="120"/>
       <port id="icadapter_dmsport_range" range="15701-15800"/>
       <process-set id="myMQApp1" restart-on-death="true" numprocs="1">
        <module-data>
          <category id="start-parameters">
              <data id="java-parameters" value="-Xms8M"/>
              <data id="class-name"
               value="oracle.oai.agent.service.AgentService"/>
          </category>
          <category id="stop-parameters">
               <data id="java-parameters" value="-mx64m"/>
               <data id="class-name"
                value="oracle.oai.agent.proxy.ShutdownAgent"/>
               <data id="application-parameters"
                value="persistence/Agent.ior"/>
          </category>
        </module-data>
       </process-set>
</process-type>
```

3. Save the `opmn.xml` file.
4. Navigate to the `MiddleTier\opmn\bin` directory and run the following command to reload the OPMN:

   ```
   opmnctl reload
   ```

5. You can start the `myMQApp1` adapter by using the following command

   ```
   opmnctl startproc ias-component="InterConnect" process-type="myMQApp1"
   ```

6. Navigate to the `MiddleTier\bin` directory and run the following command to start the Enterprise Manager:

   ```
   emctl start iasconsole
   ```

7. Login to the Oracle Enterprise Manager 10*g* Application Server Control Console to view and manage the newly installed or copied adapter. For information about how to use Oracle Enterprise Manager 10*g* Application Server Control Console , refer to the *Oracle Application Server Integration InterConnect User's Guide*

Note: While installing multiple adapters in the same computer, the copyadapter script does not create entries for the new adapter's password in the Oracle Wallet. You need to manually create a password for this new adapter using the Oracle Wallet Manager. To store the password in Oracle Wallet, use the following format:

```
ApplicationName/password
```

The number of entries is dependent on the type of adapter. For example, Database Adapter needs two entries whereas AQ Adapter needs only one entry. For more information about how to manage your passwords in Oracle Wallet, refer to Section A.6, "How do I secure my passwords?" in Appendix A, "Frequently Asked Questions"

2.3 Configuring the WebSphere MQ Adapter

After an WebSphere MQ adapter installation, you can configure it according to your requirements. The following tables describe the location and details of the configuration files.

Table 2–1 describes the location where the adapter is installed.

Table 2–1 WebSphere MQ Adapter Directory

Platform	Directory
UNIX	`ORACLE_HOME/integration/interconnect/adapters/Application`
Windows	`ORACLE_HOME\integration\interconnect\adapters\Application`

Table 2–2 describes the executable files of the WebSphere MQ adapter.

Table 2–2 Executable Files

File	Description
`start` (UNIX)	Does not use parameters; starts the adapter.
`start.bat` (Windows)	Does not use parameters; starts the adapter.
`stop` (UNIX)	Does not use parameters; stops the adapter.
`stop.bat` (Windows)	Does not use parameters; stops the adapter.

Table 2–3 describes the WebSphere MQ adapter configuration files.

Table 2–3 Configuration Files

File	Description
`adapter.ini` (UNIX)	Consists of all the initialization parameters that the adapter reads at startup.
`adapter.ini` (Windows)	Consists of all the initialization parameters that the adapter reads at startup.
`d3l-file.xml`	One or more D3L XML files that describe the mappings between WebSphere MQ native/binary fixed-structure messages and OracleAS Integration InterConnect Application View messages.

Table 2–4 describes the directories used by the WebSphere MQ adapter.

Table 2–4 Directories

File	Description
`logs`	The adapter activity is logged in subdirectories of the `logs` directory. Subdirectory names take the following form: *`timestamp_in_milliseconds`* Each time the adapter is run, a new subdirectory is created for `the log.xml` log file.
`persistence`	The messages are made available in this directory. Do not edit this directory or its files.

2.3.1 Using the Application Parameter

The WebSphere MQ adapter has a generic transformation engine that uses metadata from the repository as run-time instructions to perform transformations. The application parameter defines the capabilities of an adapter, such as the messages to be published and subscribed, and the transformations to be performed. The application parameter enables the adapter to retrieve only the relevant metadata from the repository. The application parameter must match the corresponding application name that will be defined in iStudio under the Applications folder.

If you use prepackaged metadata, then import it into the repository and start iStudio to find the corresponding application under the Applications folder. You can use this as the application name for the adapter you are installing.

2.3.2 WebSphere MQ Adapter Ini File Settings

The following `.ini` files are used to configure the WebSphere MQ adapter:

- hub.ini Parameters
- adapter.ini Parameters

2.3.2.1 hub.ini Parameters

The WebSphere MQ adapter connects to the hub database using parameters in the `hub.ini` file located in the `hub` directory. Table 2–5 lists the description and example of each parameter.

Table 2–5 hub.ini Parameters

Parameter	Description	Example
`hub_host`	The name of the computer hosting the hub database. There is no default value. The value is set during installation.	`hub_host=mpscottpc`
`hub_instance`	The SID of the hub database. There is no default value. The value is set during installation.	`hub_instance=orcl`
`hub_port`	The TNS listener port number for the hub database instance. There is no default value. The value is set during installation.	`hub_port=1521`
`hub_username`	The name of the hub database schema (or user name). The default value is `ichub`.	`hub_username=ichub`
`repository_name`	The name of the repository that communicates with the adapter. The default value is `InterConnectRepository`.	`repository_name=InterConnectRepository`

Oracle Real Application Clusters hub.ini Parameters

When a hub is installed on a Oracle Real Application Clusters database, the parameters listed in Table 2–6 represent information on additional nodes used for connection and configuration. These parameters are in addition to the default parameters for the primary node. In Table 2–6, x represents the node number which can range from 2 to the total number of nodes in a cluster. For example, if the cluster contains 4 nodes, then x can be a value between 2 and 4.

Table 2–6 Real Application Clusters Hub.ini Parameters

Parameter	Description	Example
`hub_host`*x*	The host where the Real Application Clusters database is installed.	`hub_host2=dscott13`
`hub_instance`*x*	The instance on the respective node.	`hub_instance2=orcl2`
`hub_num_nodes`	The number of nodes in a cluster.	`hub_num_nodes=4`
`hub_port`*x*	The port where the TNS listener is listening.	`hub_port2=1521`

2.3.2.2 adapter.ini Parameters

The agent component of the WebSphere MQ adapter reads the `adapter.ini` file at run time to access WebSphere MQ adapter parameter configuration information. Table 2–7 lists the description and an example of each parameter.

Table 2–7 adapter.ini Parameters

Parameter	Description	Example
`agent_admin_port`	Specifies the port through which the adapter can be accessed through firewalls. Possible value: A valid port number Default value: None	`agent_admin_port=1059`
`agent_delete_file_cache_at_startup`	Specifies whether to delete the cached metadata during startup. If any agent caching method is enabled, then metadata from the repository is cached locally on the file system. Set the parameter to `true` to delete all cached metadata on startup. Possible values: `true` or `false` Default value: `false` **Note:** After changing metadata or DVM tables for the adapter in iStudio, you must delete the cache to guarantee access to new metadata or table information.	`agent_delete_file_cache_at_startup=false`
`agent_dvm_table_caching`	Specifies the Domain Value Mapping (DVM) table caching algorithm. Possible values: ■ `startup`: Cache all DVM tables at startup. This may be time-consuming if there are many tables in the repository. ■ `demand`: Cache tables as they are used. ■ `none`: No caching. This slows down performance. Default value: `demand`	`agent_dvm_table_caching=demand`

Table 2–7 (Cont.) adapter.ini Parameters

Parameter	Description	Example
`agent_log_level`	Specifies the amount of logging necessary. Possible values: `0`=errors only `1`=status and errors `2`=trace, status, and errors Default value: `1`	`agent_log_level=2`
`agent_lookup_table_caching`	Specifies the lookup table caching algorithm. Possible values: ■ `startup`: Cache all lookup tables at startup. This may be time-consuming if there are many tables in the repository. ■ `demand`: Cache tables as they are used. ■ `none`: No caching. This slows down performance. Default value: `demand`	`agent_lookup_table_caching=demand`
`agent_max_ao_cache_size`	Specifies the maximum number of application object metadata to cache. Possible value: An integer greater than or equal to `1` Default value: `200`	`agent_max_ao_cache_size=200`
`agent_max_co_cache_size`	Specifies the maximum number of common object metadata to cache. Possible value: An integer greater than or equal to `1` Default value: `100`	`agent_max_co_cache_size=100`
`agent_max_dvm_table_cache_size`	Specifies the maximum number of DVM tables to cache. Possible value: An integer greater than or equal to `1` Default value: `200`	`agent_max_dvm_table_cache_size=200`
`agent_max_lookup_table_cache_size`	Specifies the maximum number of lookup tables to cache. Possible value: Any integer greater than or equal to `1` Default value: `200`	`agent_max_lookup_table_cache_size=200`
`agent_max_message_metadata_cache_size`	Specifies the maximum number of message metadata (publish/subscribe and invoke/implement) to cache. Possible value: An integer greater than or equal to `1` Default value: `200`	`agent_max_message_metadata_cache_size=200`
`agent_max_queue_size`	Specifies the maximum size to which internal OracleAS Integration InterConnect message queues can grow. Possible value: An integer greater than or equal to `1` Default value: `1000`	`agent_max_queue_size=1000`
`agent_message_selector`	Specifies conditions for message selection when the adapter registers its subscription with the hub. Possible value: A valid Oracle Advanced Queue message selector string (such as`'%,aqapp,%'`). Default value: None	`agent_message_selector=%,aqapp,%`

Table 2–7 (Cont.) adapter.ini Parameters

Parameter	Description	Example
`agent_metadata_caching`	Specifies the metadata caching algorithm. Possible values: ■ `startup`: Cache everything at startup. This may be time-consuming if there are many tables in the repository. ■ `demand`: Cache metadata as it is used. ■ `none`: No caching. This slows down performance. Default value: `demand`	`agent_metadata_caching=demand`
`agent_persistence_cleanup_interval`	Specifies how often to run the persistence cleaner thread in milliseconds. Possible value: An integer greater than or equal to `30000` milliseconds. Default value: `60000`.	`agent_persistence_cleanup_interval=60000`
`agent_persistence_queue_size`	Specifies the maximum size of internal OracleAS Integration InterConnect persistence queues. Possible value: An integer greater than or equal to `1` Default value: `1000`	`agent_persistence_queue_size=1000`
`agent_persistence_retry_interval`	Specifies how often the persistence thread retries when it fails to send an OracleAS Integration InterConnect message. Possible value: An integer greater than or equal to `5000` milliseconds. Default value: `60000`	`agent_persistence_retry_interval=60000`
`agent_pipeline_from_hub`	Specifies whether to activate the pipeline for messages from the hub to the bridge. If you set the pipeline to `false`, then the file persistence is not used in that direction. Possible value: `true`, `false` Default value: `false`	`agent_pipeline_from_hub=false`
`agent_pipeline_to_hub`	Specifies whether to activate the pipeline for messages from the bridge to the hub. If you set the pipeline to `false`, then the file persistence is not used in that direction. Possible value: `true`, `false`. Default value: `false`	`agent_pipeline_to_hub=false`
`agent_reply_message_selector`	Specifies the application instance to which the reply must be sent. This parameter is used if multiple adapter instances exist for the given application and given partition. Possible value: A string built using the application name (`parameter:application`) concatenated with the instance number (`parameter:instance_number`) Default value: None	`If application=aqapp, instance_number=2, then agent_reply_message_selector=recipient_list like '%,aqapp2,%'`

Table 2–7 (Cont.) adapter.ini Parameters

Parameter	Description	Example
`agent_reply_subscriber_name`	Specifies the subscriber name used when multiple adapter instances are used for the given application and given partition. This parameter is optional if only one instance is running. Possible value: The application name (`parameter:application`) concatenated with the instance number (`parameter:instance_number`) Default value: None	`If application=mpapp and instance_number=2, then agent_reply_subscriber_name=mqapp2`
`agent_subscriber_name`	Specifies the subscriber name used when this adapter registers its subscription. Possible value: A valid Oracle Advanced Queue subscriber name Default value: None	`agent_subscriber_name=mqapp`
`agent_throughput_measurement_enabled`	Specifies if the throughput measurement is enabled. Set this parameter to `true` to activate throughput measurements. Default value: `true`	`agent_throughput_measurement_enabled=true`
`agent_tracking_enabled`	Specifies if message tracking is enabled. Set this parameter to `false` to turn off tracking of messages. Set this parameter to `true` to track messages with tracking fields set in iStudio. Default value: `true`	`agent_tracking_enabled=true`
`agent_use_custom_hub_dtd`	Specifies whether to use a custom DTD for the common view message when handing it to the hub. By default, adapters use a specific OracleAS Integration InterConnect DTD for all messages sent to the hub. Set this parameter to `true` to have the adapter use the DTD imported for the message of the common view instead of the OracleAS Integration InterConnect DTD. Default value: None	`agent_use_custom_hub_dtd=false`
`application`	Specifies the name of the application to which this adapter connects. This must match the name specified in iStudio while creating metadata. Possible value: An alphanumeric string Default value: None	`application=mqapp`
`encoding`	Specifies the character encoding for published messages. The adapter uses this parameter to generate encoding information for the encoding tag of transformed OracleAS Integration InterConnect messages. OracleAS Integration InterConnect represents messages internally as XML documents. Possible value: A valid character encoding Default value: `UTF-8` When there is no existing encoding in the subscribed message, this parameter will be used to explicitly specify the encoding of the published message. This parameter will be ignored when the encoding already exists in the subscribed message.	`encoding=Shift_JIS`

Table 2–7 (Cont.) adapter.ini Parameters

Parameter	Description	Example
`external_dtd_base_url`	Specify the base URL for loading external enitites and DTDs. This specifies to the XML parser to resolve the external entities in the instance document using the given URL. Possible value: A URL Default value: The URL of the current user directory	`external_dtd_base_url=file://C:\`*`ORACLEHOME`*`\Integration\InterConnect10_1_2\adapters\MQApp\`
`instance_number`	Specifies the instance number to which this adapter corresponds. Specify a value only if you have multiple adapter instances for the given application with the given partition. Possible value: An integer greater than or equal to `1` Default value: None	`instance_number=1`
`nls_country`	Specifies the ISO country code. The codes are defined by ISO-3166. Possible value: A valid code. A full list of the codes is available at `http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html` Default value: `US` **Note**: This parameter specifies date format and is applicable only for the date format.	`nls_country=US`

`nls_date_format`	Specifies the format for a date field expressed as a string.
	Possible value: A valid date format pattern as shown in Table 2–8 for the definitions of the format characters.
	Default value: `EEE MMM dd HHmmss zzz yyyy`
		Date format pattern `dd/MMM/yyyy` can represent 01/01/2003.
		`nls_date_format=dd-MMM-yy`
		Multiple date formats can be specified as `num_nls_formats=2`
		`nls_date_format1=dd-MMM-yy`
		`nls_date_format2=dd/MMM/yy`

`nls_language`	Specifies the ISO language code. The codes are defined by ISO-639. Possible value: A valid code. A full list of these codes is available at `http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt` Default value: `en` **Note**: This parameter specifies date format and is applicable only for the date format.	`nls_language=en`
`partition`	Specifies the partition this adapter handles as specified in iStudio. Possible value: An alphanumeric string Default value: None	`partition=germany`

Table 2–7 (Cont.) adapter.ini Parameters

Parameter	Description	Example
`service_class`	Specifies the entry class for the Windows service. Possible value: `oracle/oai/agent/service/AgentService` Default value: None	`service_class=oracle/oai/agent/service/AgentService`
`service_classpath`	Specifies the class path used by the adapter JVM. If a custom adapter is developed and the adapter is to pick up any additional jar files, then add the files to the existing set of jar files. Possible value: A valid `PATH` setting Default value: None This parameter is only for Microsoft Windows.	`service_classpath=D:\oracle\oraic\integration\interconnect\lib\oai.jar;D:\oracle\oraic\jdbc\classes12.zip`
`service_jdk_dll`	Specifies the Dynamic Link Library (DLL) that the adapter JVM should use. Possible value: A valid `jvm.dll` Default value: `jvm.dll` This parameter is only for Microsoft Windows.	`service_jdk_dll=jvm.dll`
`service_jdk_version`	Specifies the JDK version that the adapter JVM should use. Possible value: A valid JDK version number Default value: 1.4 This parameter is only for Microsoft Windows.	`service_jdk_version=1.4`
`service_max_heap_size`	Specifies the maximum heap size for the adapter JVM. Possible value: A valid JVM heap size Default value: `536870912` This parameter is only for Microsoft Windows.	`service_max_heap_size=536870912`
`service_max_java_stack_size`	Specifies the maximum size to which the JVM stack can grow. Possible value: A valid JVM maximum stack size Default value: Default value for the JVM This parameter is only for Microsoft Windows.	`service_max_java_stack_size=409600`
`service_max_native_stack_size`	Specifies the maximum size to which the JVM native stack can grow. Possible value: A valid JVM maximum native stack size Default value: Default value for the JVM This parameter is only for Microsoft Windows.	`service_max_native_size=131072`
`service_min_heap_size`	Specifies the minimum heap size for the adapter JVM. Possible value: A valid JVM heap size Default value: `536870912` This parameter is only for Microsoft Windows.	`service_min_heap_size=536870912`

Table 2–7 (Cont.) adapter.ini Parameters

Parameter	Description	Example
`service_num_vm_args`	Specifies the number of `service_vm_arg`*`number`* parameters specified in JVM. Possible value: The number of `service_vm_arg`*`number`* parameters Default value: None This parameter is only for Microsoft Windows.	`service_num_vm_args=1`
`service_path`	Specifies the environment variable `PATH`. The `PATH` variable is set before starting the Java Virtual Machine (JVM). Typically, list all directories that contain necessary DLLs. Possible value: A valid `PATH` environment variable setting Default value: None This parameter is only for Microsoft Windows.	`service_path=%JREHOME%\bin;D:\oracle\oraic\bin`

`service_vm_arg`*`number`*	Specifies any additional arguments to the JVM. For example, to retrieve line numbers in any stack traces, set `service_vm_arg1=java.compiler=NONE`. If a list of arguments exists, then use multiple parameters as shown in the example, by incrementing the last digit by `1`.
	Possible value: Valid JVM arguments
	Default value: None
	This parameter is only for Microsoft Windows.
		`service_vm_arg1=java.compiler=NONE`
		`service_vm_arg2=oai.adapter=.aq`

Table 2–8 shows the reserved characters used to specify the value of the `nls_date_format` parameter. Use these characters to define date formats.

Table 2–8 Reserved Characters for the value of the nls_date_format Parameter

Letter	Description	Example
`G`	Era designator	`AD`
`y`	Year	`1996` or `96`
`M`	Month in year	`July` or `Jul` or `07`
`w`	Week in year	`27`
`W`	Week in month	`2`
`D`	Day in year	`189`
`d`	Day in month	`10`
`F`	Day of week in month	`Number 2`
`E`	Day in week	`Tuesday` or `Tue`
`a`	a.m./p.m. marker	`P.M.`
`H`	Hour in day (0-23)	`0`
`k`	Hour in day (1-24)	`24`
`K`	Hour in a.m./p.m. (0-11)	`0`
`h`	Hour in a.m./p.m. (1-12)	`12`
`m`	Minute in hour	`30`

Table 2–8 (Cont.) Reserved Characters for the value of the nls_date_format Parameter

Letter	Description	Example
`s`	Second in minute	`55`
`S`	Millisecond	`978`

2.3.2.3 WebSphere MQ Adapter-specific Parameters

Table 2–9 lists the parameters specific to the WebSphere MQ adapter.

Table 2–9 WebSphere MQ Adapter-specific Parameters

Parameter	Description	Example
`bridge_class`	Specifies the entry class for the WebSphere MQ adapter. A value must be specified and cannot be modified later. Possible value: `oracle.oai.agent.adapter.technology.TechBridge`. Default value: None	`bridge_class=oracle.oai.agent.adapter.mq.MQBridge`
`Encrypted_mq.default.password`	Specifies the WebSphere MQ (encrypted) password when connecting to the queue manager. Equivalent to the WebSphere MQ environment variable MQ_PASSWORD. The value may be used to verify the identity of the WebSphere MQ adapter. Default value: None **Note**: All passwords are stored in Oracle Wallet. Refer to Section A.6, "How do I secure my passwords?" in Appendix A, "Frequently Asked Questions" for more details on how to modify and retrieve the password using Oracle Wallet.	`Encrypted_mq.default.password=112411071071106510801094108410731070107110811069`
`mq.default.connection_type`	Specifies the type of connection to make to an WebSphere MQ queue manager. Possible values: bind (local) or client (remote) Default value: None	`mq.default.connection_type=client`
`mq.default.receiver.durable`	Defines whether or not a durable subscriber should be used to subscribe to the topic. This is used only if the `receiver.destination.uri` parameter specifies a JMS topic. Possible value: Y or N Default value is N	`mq.default.receiver.durable=Y`
`mq.default.receiver.transacted`	Specifies whether or not the JMS sessions for the receive URI should be transacted. The JMS session for the sender URI is always transacted. Possible value: Y or N Default value: N	`mq.default.receiver.transacted=Y`

Table 2–9 (Cont.) WebSphere MQ Adapter-specific Parameters

Parameter	Description	Example
`mq.default.sender.seq_queue.uri`	Specifies a URI for the WebSphere MQ transaction id (sequence generator) queue used during send transactions. It can refer to the same queue as `mq.default.sender.log_queue.uri`. Possible values: A JMS queue URI Default value: None	`mq.default.sender.log_queue.uri=queue:///OIA.SEQ.QUEUE`
`mq.default.ccsid`	Specifies the coded-character-set-ID in use on connections instead of the default. Possible values: Refer to table 16 in the WebSphere MQ Using Java Guide Default value: blank (~819)	`mq.default.ccsid=1208`
`mq.default.channel`	Specifies the name of the WebSphere MQ channel to use for the client connection. Possible value: Any valid WebSphere MQ channel name Default value: None	`mq.default.channel=SYSTEM.DEF.SVRCONN`
`mq.default.event.name`	Specifies the default event name. This parameter should be used if the bridge will only handle one single fixed event name for outbound messages (from WebSphere MQ) and none of the other options are feasible to use. This parameter requires only one D3L file defined, with an event name exactly matching this hardcoded event name. Possible value: A valid OracleAS Integration InterConnect event name Default value: None	`mq.default.event.name=Price.update`
`mq.default.event.property`	Defines the default event property. If the sending external application is able to specify the event name as a message property value, then use this parameter to define the name of the message property that will carry the message event name. Possible value: A valid JMS message property name Default value: None	`mq.default.event.property=MyApp_OAIEventProperty`
`mq.default.event.exit`	Allows a custom Java class to be defined to determine which event name the native WebSphere MQ message corresponds to. It is invoked by the bridge, which provides the received JMS message as input, expecting the event name in return (as a String). This Java class must implement the `oracle.oai.agent.adapter.mqseries.MQEventExit` interface. Possible value: The Java class name of a class that implements the `oracle.oai.agent.adapter.mqseries.MQEventExit` interface. Default value: None	`mq.default.event.exit=mypackage.myMqEventExit`

Table 2–9 (Cont.) WebSphere MQ Adapter-specific Parameters

Parameter	Description	Example
`mq.default.event.use_mq_fmt`	Specifies the usage of the IBM WebSphere MQ Message Format field. If this parameter value is Y, then the bridge uses the IBM WebSphere MQ Message Format field as the name of the OracleAS Integration InterConnect event. This message field or property is often referred to as: ■ `(C)—MQMD Format field (MQFMT)` ■ `(Java)—com.ibm.mq.jms.JMSC.FORMAT_PROPERTY` Possible values: `Y` or `N` Default value: `N`	`mq.default.event.use_mq_fmt=Y`
`mq.default.hostname`	Specifies the DNS name of the host where the queue manager resides. Possible value: A valid hostname that can be reached over the network from the WebSphere MQ adapter. Default value: None	`mq.default.hostname=mqsvrhost1.acme.com`
`mq.default.polling_interval`	Specifies the number of milliseconds between attempts to receive a message. Possible value: 0-`java.lang.Long.MAX_VALUE` Default value: `5000`	`mq.default.polling_interval=5000`
`mq.default.port`	Specifies the port to connect to on the WebSphere MQ Server host (IBM's default is 1414). Possible value: A valid port number for the WebSphere MQ listener Default value: None	`mq.default.port=1414`
`mq.default.queue_manager`	Specifies the name of the WebSphere MQ queue manager to connect to. Possible value: Any WebSphere MQ queue manager name Default value: None	`mq.default.queue_manager=mars.queue.manager`
`mq.default.receive_exit`	Specifies the fully qualified class name of the receive exit being used. Possible value: The classname of a Java class that implements `com.ibm.mq.MQReveiveExit` Default value: None	`mq.default.receive_exit=mypackage.myReceiveExit`
`mq.default.receiver.destination.uri`	Specifies a URI for the WebSphere MQ outbound queue or topic from which messages will be received. Used for listening to incoming messages or as a ReplyTo address when sending request messages to WebSphere MQ. Possible values: A JMS queue URI Default value: None	`mq.default.receiver.destination.uri=topic://SAP/Events/HR/newEmployee`

Table 2–9 (Cont.) WebSphere MQ Adapter-specific Parameters

Parameter	Description	Example
`mq.default.receiver.selector`	Specifies the JMS selector expression applied while dequeueing from the receiver destination. Possible values: A JMS selector expression Default value: None	`mq.default.receiver.selector=JMS_IBM_Format <> 'MQSTR' AND JMSXUserID = 'scott'`
`mq.default.receiver.exception.uri`	Specifies a URI for an WebSphere MQ queue where faulty native messages will be placed. Default value: None.	`mq.default.receiver.exception.uri=queue:///EXCEPTION.QUEUE`
`mq.default.security_exit`	Specifies the fully qualified class name of the security exit being used. Possible value: The classname of a Java class that implements `com.ibm.mq.MQSecurityExit` Default value: None	`mq.default.security_exit=mypackage.MySecurityExit`
`mq.default.send_exit`	Specifies the fully qualified class name of the send exit being used. Possible value: The classname of a Java class that implements `com.ibm.mq.MQSendExit` Default value: None	`mq.default.send_exit=mypackage.mySendExit`
`mq.default.sender.destination.uri`	Specifies the URI for the WebSphere MQ inbound queue to which messages will be sent from OracleAS Integration InterConnect. Possible values: A JMS queue URI Default value: None	`mq.default.sender.destination.uri=queue:///INBOUND.QUEUE?priority=1`
`mq.default.sender.log_queue.uri`	Specifies a URI for the WebSphere MQ log queue used during send transactions. Possible values: A JMS queue URI Default value: None	`mq.default.sender.log_queue.uri=queue:///OAI.LOG.QUEUE`
`mq.default.sender.mqfmt`	Supresses the JMS specific header information. The WebSphere MQ adapter will normally read and write JMS messages from and to WebSphere MQ queues, which include a JMS specific header section. To suppress this header when interacting with external non-JMS clients (C or non-JMS Java applications), define this property. It will also defines the message `MQMD` Format field of each message being sent by the adapter. If the value is set to `MQFMT_STRING`, then it will cause all messages to be sent as Text messages, even in D3L mode. Normally, D3L mode will cause the adapter to send only `Bytes` messages. Default value: None	`mq.default.sender.mqfmt=MQFMT_STRING`

Table 2–9 (Cont.) WebSphere MQ Adapter-specific Parameters

Parameter	Description	Example
`mq.default.trans_id_expiry`	Specifies the number of milliseconds before an idle transaction identifier will expire. Possible value: 0-`java.lang.Long.MAX_VALUE` Default value: `60000`	`mq.default.trans_id_expiry=360000`
`mq.default.user`	Specifies the WebSphere MQ user ID when connecting to the queue manager. Equivalent to the WebSphere MQ environment variable MQ_USER_ID. The value may be used to verify the identity of the WebSphere MQ adapter. Possible value: A valid WebSphere MQ user name Default value: None	`mq.default.user=mqm`
`ota.type`	Defines the type of payload this adapter handles. Possible values: `XML` and `D3L` Default value: `XML`	`ota.type=D3L`

2.4 Uninstalling the WebSphere MQ Adapter

To uninstall the WebSphere MQ adapter, perform the following:

1. Navigate to the *`MiddleTier`*`\opmn\bin` directory.
2. Run the following command to check the adapter status.

   ```
   opmnctl status
   ```

3. If the WebSphere MQ adapter instance that you want to remove is running, stop it by using the the following command:

   ```
   opmnctl stopproc ias-component="InterConnect" process-type="MQApp"
   ```

   where `MQApp` is the name of the WebSphere MQ adapter instance.
4. Navigate to the *`MiddleTier`*`\bin` directory and run the following command to stop the Enterprise Manager:

   ```
   emctl stop iasconsole
   ```

5. Carefully, remove the adapter process-type entry from the `opmn.xml` file located in the *`MiddleTier`*`\opmn\conf` directory. For example, to remove an WebSphere MQ adapter instance `myMQApp1`, delete the following information specific to the adapter instance:

   ```
   <process-type id="MyMQApp1" module-id="adapter" working-dir="$ORACLE_
   HOME/integration/interconnect/adapters/MyMQApp1" status="enabled">
          <start timeout="600" retry="2"/>
          <stop timeout="120"/>
          <port id="icadapter_dmsport_range" range="15701-15800"/>
          <process-set id="MyMQApp1" restart-on-death="true" numprocs="1">
               <module-data>
                   <category id="start-parameters">
                       <data id="java-parameters" value="-Xms8M"/>
                       <data id="class-name"
                        value="oracle.oai.agent.service.AgentService"/>
   ```

```
                </category>
                <category id="stop-parameters">
                    <data id="java-parameters" value="-mx64m"/>
                    <data id="class-name"
                     value="oracle.oai.agent.proxy.ShutdownAgent"/>
                    <data id="application-parameters"
                     value="persistence/Agent.ior"/>
                </category>
            </module-data>
       </process-set>
</process-type>
```

6. Save the `opmn.xml` file.
7. Navigate to the *`MiddleTier`*`\opmn\bin` directory and run the following command to reload the OPMN:

   ```
   opmnctl reload
   ```

8. Navigate to the *`ORACLE_HOME`*`\integration\interconnect\adapters` directory and delete the folder that was created for the removed adapter instance.
9. Navigate to the *`MiddleTier`*`\bin` directory and run the following command to start the Enterprise Manager:

   ```
   emctl start iasconsole
   ```

3 Design-Time and Run-Time Concepts

This chapter describes the design-time and run-time concepts for the WebSphere MQ adapter. It contains the following topics:

- WebSphere MQ Adapter Design-Time Concepts
- WebSphere MQ Adapter Run-Time Concepts
- Starting the WebSphere MQ Adapter
- Stopping the WebSphere MQ Adapter

3.1 WebSphere MQ Adapter Design-Time Concepts

The WebSphere MQ adapter can handle XML and D3L structured payloads, such as pure XML data with strings beginning with `<xml...`, and binary data described by a D3L XML file.

3.1.1 XML Payload

You can import a Document Type Definition (DTD) or XML Schema Definition (XSD) in iStudio, which determines how the WebSphere MQ adapter parses a received XML document into an OracleAS Integration InterConnect application view event. In addition, you can use the XSD or DTD to describe how an inbound application view message is converted to an XML document. Use the message type option XML when defining a new integration point in any of the event wizards.

Ensure that the `ota.type` parameter in the `adapter.ini` file is set to `XML`, instead of `D3L`.

When the WebSphere MQ adapter operates in the XML payload mode, no transformations are performed on the messages between native view and application view. Any Extensible Stylesheet Language Transformations (XSLT) should be performed either before sending an XML document to OracleAS Integration InterConnect, or after receiving one from OracleAS Integration InterConnect.

3.1.2 D3L Payload

The WebSphere MQ adapter performs a two-way conversion and transformation of messages between application view and native format.

An application based on the WebSphere MQ adapter can use the iStudio Message Type D3L and the iStudio D3L Data Type Import options when importing a data type. In this case, messages received or sent by the WebSphere MQ adapter must adhere to the fixed byte-level layout defined in a D3L XML file.

The D3L Data Type Import option can also define common view datatypes.

> **See Also:** *Oracle Application Server Integration InterConnect User's Guide*, Appendix B, for additional information on D3L

3.2 WebSphere MQ Adapter Run-Time Concepts

This section describes the key run-time components of the WebSphere MQ adapter. It contains the following topics:

- How the WebSphere MQ Adapter Works
- Support for Request-Reply in D3L Mode

3.2.1 How the WebSphere MQ Adapter Works

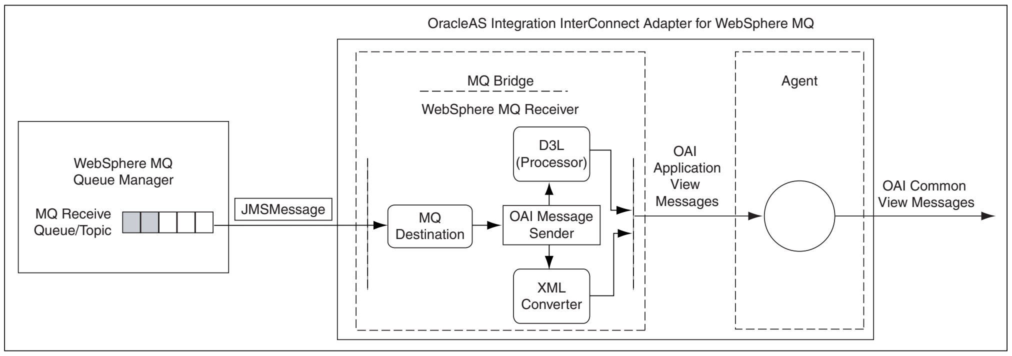
This section gives an overview of how the WebSphere MQ adapter works. It contains the following topics:

- Outbound
- D3L Disambiguation
- Inbound

3.2.1.1 Outbound

The WebSphere MQ adapter is comprised of the bridge and the run-time agent. The bridge constantly polls the queue chosen for publishing messages in the WebSphere MQ outbound queue. A new message in this queue indicates a new outbound OracleAS Integration InterConnect message waiting to be sent by the adapter. The adapter picks up the message, builds the corresponding OracleAS Integration InterConnect message, persists it, transforms it to the common view, and routes it to the hub. From the hub, the message is routed to the suitable subscriber.

Figure 3–1 Outbound Message Routing



The relevant parameters in `adapter.ini` pertaining to the outbound WebSphere MQ endpoint are `mq.default.receiver.*` and `mq.default.event.*`.

> **See Also:** Chapter 2, "Installation and Configuration"

3.2.1.2 D3L Disambiguation

If the `ota.type` parameter is set to `D3L`, then the WebSphere MQ bridge uses the D3L processor to parse from native or byte format to an OracleAS Integration InterConnect message object, which then is handed over to the agent as an application view event.

When the WebSphere MQ adapter receives a message from the outbound WebSphere MQ queue while operating in D3L mode, the message is construed as an sequence of bytes. The processe of determining the OracleAS Integration InterConnect event and the D3L to which this message corresponds is called D3L Disambiguation.

The WebSphere MQ adapter has six methods to determine this through a combination of header values found in the configured D3L files and the value of one of the `mq.default.event.*` parameters in the `adapter.ini` file. These methods are described as follows.

Note: The term *event name* as used in this section implies a specification of the OracleAS Integration InterConnect business object as part of the event name, prefixed followed by a dot, for example, `Order.getStatus`. The event name also synonymously includes OracleAS Integration InterConnect procedure names.

3.2.1.2.1 D3L Disambiguation Order

The disambiguation methods are tried in the following order:

1. If only one D3L is specified in the `ota.d3ls` parameter, then it is always used.
2. Using a D3L Header and Value Pair
3. Using D3L Magic
4. Using the mq.default.event.name Parameter
5. Using the mq.default.event.use_mq_fmt Parameter
6. Using the mq.default.event.property Parameter
7. Using the mq.default.event.exit Parameter
8. Trying All D3Ls Until One Works

3.2.1.2.2 Using the mq.default.event.name Parameter

Using this parameter is the most primitive mode of operation. Using a hard-coded event name for all outbound messages received from WebSphere MQ is one example.

Example: `mq.default.event.name=Employee.updateInfo`

This example requires that exactly one D3L file has the following header:

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE message SYSTEM "d3l.dtd">
<message name="updateInfo" object="Employee" type="...
...
```

3.2.1.2.3 Using the mq.default.event.property Parameter

Use this method if the sending WebSphere MQ application can inform the WebSphere MQ adapter about which event a message corresponds to, by setting a specified message property to a given value.

To use this method, complete the following:

1. Set the `mq.default.event.property` parameter to the name of the message property that will contain the native event name.
2. Define one D3L XML for each possible value of this message property, binding the D3L file to a given value of the message property through the use of the D3L header attributes name and object.

Example: `mq.default.event.property=SAP_EvNm`

This property will accept the two distinct values `Order.evtPut` and `Order.evtGet`. Considering this, the following two D3L files should be defined:

- `sap_put.xml`

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE message SYSTEM "d3l.dtd">
<message name="evtPut" object="Order" type="...">
...
```

- `sap_get.xml`

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE message SYSTEM "d3l.dtd">
<message name="evtGet" object="Order" type="...">
...
```

Set the `ota.d3ls` parameter to `sap_put.xml,sap_get.xml`.

The `name` and `object` headers should correspond to the associated OracleAS Integration InterConnect event and business object names.

3.2.1.2.4 Using a D3L Header and Value Pair The WebSphere MQ adapter supports D3L disambiguation using the header and value attributes. For the WebSphere MQ adapter, transport message headers correspond to the WebSphere MQ message properties. Consequently, transport message header values are identical to WebSphere MQ message property values.

> **See Also:** *Oracle Application Server Integration InterConnect User's Guide,*

3.2.1.2.5 Using the mq.default.event.use_mq_fmt Parameter This mode enables the WebSphere MQ message format property to be used to select the corresponding event name. This property is often referred to as the following:

- The MQMD Format field, `MQFMT`
- In Java, `com.ibm.mq.jms.JMSC.FORMAT_PROPERTY`

For example:

Assume the `MQFMT` field of a received message from WebSphere MQ has the value `Cus.new`.

This requires the following `adapter.ini` file setting:

- `mq.default.event.use_mq_fmt=Y`

and the following D3L file:

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE message SYSTEM "d3l.dtd">
<message name="new" object="Cus" type="..."
...
```

Optionally, if the values in the `MQFMT` field do not easily map into the OracleAS Integration InterConnect event names, then you can define a the `mqfmt2event.ini` mapping file in the same directory where `adapter.ini` file is located. If this file is present, then the adapter will read the file and apply the specified event name mappings when a message is received. The format of the file is as follows:

```
<MQMFMT-field-value-1>=<OAI-business-object-name1>.<OAI-event-name1>
```

```
<MQMFMT-field-value-2>=<OAI-business-object-name2>.<OAI-event-name2>
...
<MQMFMT-field-value-n>=<OAI-business-object-namen>.<OAI-event-namen>
```

Example

```
CustNew=Customer.createCustomer
CustUpd=Customer.updateCustomer
```

Note: More than one MQMFT field value can map to the same event name.

Note: The business object and event names on the right hand side of the equal sign in the `mqfmt2event.ini` file must be matched by corresponding name and object attribute values in the associated D3L files.

3.2.1.2.6 Using the mq.default.event.exit Parameter This event name resolution method enables a Java class call-out to be registered, which is given a reference to the received JMS message. In return, the Java class call-out must tell the bridge the event name corresponding to the message. The Java class must implement the `oracle.oai.agent.adapter.mqseries.MQEventExit` interface, which has the following signature:

```
public interface MQEventExit
{
    public String getEventName(javax.jms.Message jmsMessage)
        throws oracle.oai.agent.adapter.mqseries.MQBridgeException;
```

- Example: `myEventExit.java`

```
import oracle.oai.agent.adapter.mqseries.MQBridgeException;
public class myEventExit
    implements oracle.oai.agent.adapter.mqseries.MQEventExit
{
    public String getEventName(Message jmsMessage)
       throws MQBridgeException
    {
        try
        {
            if (jmsMessage instanceof TextMessage)
            {
                String body  = ((TextMessage)jmsMessage).getText();
                String bizObj = body.substring(1,10);
                String event  = body.substring(21,30);
                return bizObj + "." + event;
            }
             else
                throw new MQBridgeException("Wrong message type");
        }
        catch (Exception e) {
            throw new MQBridgeException("Error", e);
        }
    }
}
```

3.2.1.2.7 Using D3L Magic

The D3L syntax enables a magic header attribute to be specified. If specified, the header corresponds to a sequence of bytes, specified in UTF-8 bytes, hexadecimal, or octal, that should occur at the very beginning of the native-format message. If the magic attribute in one of the registered D3L files (defined in the `ota.d3ls` parameter) matches the bytes at the beginning of the native message, then that D3L header name and object attributes are chosen as the event name.

Example: `prod_getprice.xml`

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE message SYSTEM "d3l.dtd">
<message name="getPrice" object="Product" type="..."
    magic="SYSPR01GETPRC"
...
```

If the byte stream of a received message begins with the characters `SYSPR01GETPRC`, then the event is resolved as `Product.getPrice` and the shown D3L file is subsequently used to transform the native byte message into an OracleAS Integration InterConnect Message Object.

If the magic value does not reside at the very beginning of the message, then its starting position can be offset by using the D3L `message` element attribute `startsat`.

For example:

```
<message name = "getPrice" magic="SYSPR01GETPRC" startsat="18" ...>
```

3.2.1.2.8 Trying All D3Ls Until One Works

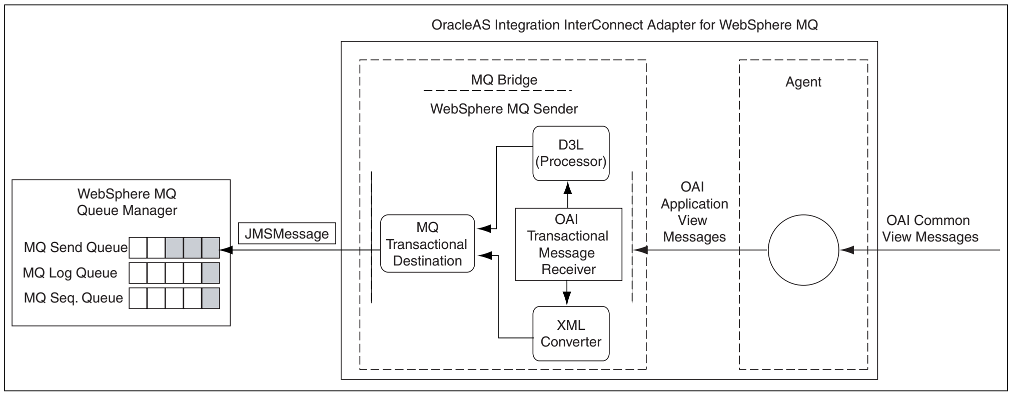
If any of the preceding methods fail, then the WebSphere MQ adapter falls back to a trial-and-error resolution scheme where each registered D3L file is tried until one succeeds. This means applying all files in the order they are listed in the `ota.d3ls` parameter in the `adapter.ini` file. If none of the D3L files succeed, then the entire D3L disambiguation process for a given message will terminates and an error message is logged. The failed message is saved in the directory where the `adapter.ini` file is located, under a name such as `MQ.FailedMsg.`*`message-id`*.

Note: The adapter subscribing to an event should be started before any other adapter can publish that event. If you publish an event before starting the subscribing adapter, then the event would not be delivered to the subscribing adapter.

3.2.1.3 Inbound

The WebSphere MQ adapter only supports sending to a single WebSphere MQ inbound endpoint, as shown in Figure 3–2.

Figure 3–2 Inbound Message Routing


OracleAS Integration InterConnect Adapter for WebSphere MQ
MQ Bridge
WebSphere MQ Sender
Agent
D3L (Processor)
WebSphere MQ Queue Manager
MQ Send Queue
MQ Log Queue
MQ Seq. Queue
JMSMessage
MQ Transactional Destination
OAI Transactional Message Receiver
OAI Application View Messages
OAI Common View Messages
XML Converter

The `mq.default.sender.*` parameter in the `adapter.ini` file pertains to the default inbound WebSphere MQ endpoint.

3.2.2 Support for Request-Reply in D3L Mode

The WebSphere MQ adapter can publish or subscribe any event and invoke or implement any procedure.

The support for invoke and implement messages, such as Procedure calls, is enabled by the native support for request and reply messages in WebSphere MQ, including its message correlation capability. It is only available when the WebSphere MQ adapter operates in D3L mode.

For request/reply scenario, some additional steps must be performed during configuration, including modifying the D3L files and defining correlation fields in iStudio.

The following instructions are based on a small example:

- Business Object: `Product`
- Procedure: `getPrice`
- Input parameters: `ProductID` and `CustomerID` as `integers`.
- Output parameters: `ProductID` as an `integer` and `Price` and `Discount` as `floats`.

These data types must be defined in two separate D3L files, one defining the native input (request) data structure, and one defining the native output (reply) data structure. The following two D3L files could serve this purpose.

3.2.2.1 getPriceIn.xml

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE message SYSTEM "d3l.dtd">
<message type="getPriceInput" name="getPrice" object="Product">

    <!-- ID type -->
    <unsigned4 id="ID" endian="little" />

    <struct id="getPriceInput">

        <field name="ProductID">     <typeref type="ID" />
    </field>
        <field name="CustomerID">    <typeref type="ID" />
    </field>
    </struct>

</message>
```

3.2.2.2 getPriceOut.xml

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE message SYSTEM "d3l.dtd">

<message type="getPriceOutput" name="getPrice" object="Product" reply="Y">

    <!-- ID type -->
    <unsigned4 id="ID" endian="little" />

    <!-- Float, as decimal number format enclosed by '$' -->
    <number id="Float"><limstring delimiter="$" /></number>

    <struct id="getPriceOutput">
        <field name="ProductID">   <typeref type="ID" />     </field>
        <field name="Price">       <typeref type="Float" />  </field>
        <field name="Discount">    <typeref type="Float" />  </field>
    </struct>

</message>
```

It is assumed that the **partner** application will be based on the Database adapter.

3.2.2.3 Invoking the Product.getPrice Procedure Using the WebSphere MQ Adapter

To invoke a procedure using the WebSphere MQ adapter in iStudio:

1. Right-click Invoked Procedures for the WebSphere MQ application and select **New**. The Invoke Wizard - Select a Procedure page is displayed.
2. Select **getPrice** as the Application.
3. Set the **Message Type** to D3L.
4. Click **Next**. The Define Application View page is displayed.
5. Click **Import** and select **D3L**.
6. Select the `getPriceIn`.xml file and mark as it as IN.
7. Select the `getPriceOut.xml` file and mark as it as OUT.
8. Click **OK**, and then click **Finish**.
9. Change to the following directory and copy the two XML files (`get*.xml`) to this directory.

Platform	Action
UNIX	*ORACLE_HOME*`/integration/interconnect/adapters/<mqapp>`
Windows	*ORACLE_HOME*`\integration\interconnect\adapters\<mqapp>`

10. List the two XML file names in the ota.d3ls parameter in the `adapter.ini` file, for example:

```
ota.d3ls=getPriceIn.xml,getPriceOut.xml
```

11. Mark the `getPriceOut.xml` D3L file as the REPLY. The WebSphere MQ adapter does not allow two D3L files defining the same business object and event name. Use the D3L message element attribute reply as follows:

```
<message type="getPriceOutput" name="getPrice" object="Product" reply="Y">
```

12. Decide and configure the D3L disambiguation scheme that enables the WebSphere MQ adapter to correctly select the `getPriceIn.xml` D3L file when it reads an outbound message from WebSphere MQ, using header/value disambiguation. For example:

```
<message type="getPriceInput" name="getPrice" object=
"Product" header="D3Lselector" value="getprice">
```

3.2.2.3.1 In (native) Invoking Application (JMS example)

```
 // This 3rd party application will send a REQUEST message to
// OAI (Invoke role), and then await a REPLY.
    BytesMessage reqMessage = session.createBytesMessage();
    byte[] getPriceMsg = new byte[] { 20, 0, 0, 0, 10, 0, 0, 0 };
    reqMessage.writeBytes(nativeBytes, 0, nativeBytes.length);
    reqMessage.setJMSReplyTo((Destination)replyQueue);
    reqMessage.setStringProperty("D3Lselector", "getprice");
    reqMessage.setIntProperty("JMS_IBM_MsgType", (int)1); //
REQUEST
    // Send REQUEST
    queueSender.send(reqMessage);
    session.commit();
    ...
    // Await REPLY
    Message replyMessage = queueReceiver.receive();
    if (replyMessage instanceof BytesMessage)
    {
        if (replyMessage.getJMSCorrelationID().
            equals(reqMessage.getJMSMessageID()))
            // Got my reply back!
```

3.2.2.3.2 In (PL/SQL) Implementing Application

```
PROCEDURE getprice(productID  IN OUT INTEGER,
                   customerID IN     INTEGER,
                   price         OUT NUMBER,
                   discount      OUT NUMBER)
IS
BEGIN
  -- Just return something
  price := 1499.95;
  discount := 10.0;
END;
```

Which gets invoked from the stub generated by iStudio:

```
PACKAGE BODY Product AS
    PROCEDURE imp_getPrice_QA_V1(io_PRODUCTID IN OUT NUMBER,
                                 i_CUSTOMERID IN     NUMBER,
                                 o_PRICE         OUT NUMBER,
                                 o_DISCOUNT      OUT NUMBER)
AS
BEGIN
```

```
    getprice(io_PRODUCTID, i_CUSTOMERID, o_PRICE, o_DISCOUNT);
END imp_getPrice_QA_V1;
```

3.2.2.4 Implementing Product.getPrice Procedure Using the WebSphere MQ Adapter

To implement a procedure using the WebSphere MQ adapter in iStudio:

1. Right-click **Implemented Procedures** for the WebSphere MQ application and select **New**. The Implement Wizard - Select a Procedure page is displayed.
2. Select **getPrice** as the Application.
3. Set the Message Type to **D3L** and click **Next**. The Define Application View page is displayed.
4. Click **Import** and select **D3L**.
5. Select the `getPriceIn.xml` file and mark it as IN.
6. Select the `getPriceOut.xml` file and mark it as OUT.
7. Click **OK**. The Define Correlation Fields page is displayed.
8. Select the two fields in the Input and Output data structures. These fields are used to correlate a response to its original request.
9. Click **OK** and then click **Finish**.
10. Change to the following directory and copy the two XML files (`get*.xml`) to this directory.

Platform	Action
UNIX	*ORACLE_HOME*`/integration/interconnect/adapters/<mqapp>`
Windows	*ORACLE_HOME*`\integration\interconnect\adapters\<mqapp>`

11. List the two XML file names in the ota.d3ls parameter in the `adapter.ini` file, for example:

    ```
    ota.d3ls=getPriceIn.xml,getPriceOut.xml
    ```

12. Mark the `getPriceOut.xml` D3L file as the REPLY. The WebSphere MQ adapter does not allow two D3Ls defining the same BusinessObject and EventName. Use the D3L message element attribute reply, as follows:

    ```
    <message type="getPriceOutput" name="getPrice" object="Product" reply="Y">
    ```

13. Decide and configure the D3L disambiguation scheme that enables the WebSphere MQ adapter to correctly select the `getPriceOut.xml` D3L file when it reads an outbound message from WebSphere MQ. The following example uses header/value disambiguation:

    ```
    <message type="getPriceOutput" name="getPrice" object="Product" reply="Y"
    header="D3Lselector" value="getpricereply">
    ```

3.2.2.4.1 In (Native) Implementing (or Invoked) Application (JMS Example)

```
// This 3rd party application will consume/read a REQUEST message from
```

```
// OAI (Implement role), and return a REPLY.

// Read REQUEST
Message reqMessage = queueReceiver.receive();

if (reqMessage instanceof BytesMessage)
{
    // Extract ProductID from request
    byte[] productID = new byte[4];
    ((BytesMessage)reqMessage).readBytes(productID);

    // Construct reply (binary lay-out message)
    byte[] getPriceReply = new byte[] {
        0, 0, 0, 0,                        // Product ID
        '$', '2','0','0','.','7','5','$',  // Price
        '$',     '1','5','.','1','0','$'   // Discount
    };

    // Copy the Product ID received in Request into the Reply
    // so OAI can correlate the reply to the original request.
    for (int i = 0; i < 4; i++)
        getPriceReply[i] = productID[i];
    ....

    BytesMessage replyMessage = session.createBytesMessage();
    replyMessage.writeBytes(getPriceReply, 0, getPriceReply.length);

        replyMessage.setJMSCorrelationID(reqMessage.getJMSMessageID());
        replyMessage.setIntProperty("JMS_IBM_MsgType", (int)2); // REPLY
        replyMessage.setStringProperty("D3Lselector", "getpricereply");

    // Send REPLY
    queueSender.send(replyMessage);
    session.commit();
```

3.2.2.4.2 In (PL/SQL) Invoking Application (Asynchronously)

```
-- Invoking procedure
PROCEDURE INVGETPRICE(prodID IN NUMBER, custID IN NUMBER)
AS
    moid  NUMBER;
    aoid  NUMBER;
    naoid NUMBER;
BEGIN
    Product.crMsg_getPrice_QA_V1(moid, aoid);
    naoid := Product.cr_getPriceInput_getPriceInput(prodID, custID, moid, aoid);
    Product.inv_getPrice_QA_V1(moid,'DBAPP');
END;
```

When OracleAS Integration InterConnect receives a reply from the WebSphere MQ application, it invokes a procedure, for example:

```
PROCEDURE sub_getPrice_QA_V1(getPriceOutput IN dbapp_getPriceOutput_QA_V1)
AS
BEGIN
    -- Save Reply
    INSERT INTO price_reply (prodid, price, discount)
    VALUES (getPriceOutput.ProductID,
        getPriceOutput.Price,
        getPriceOutput.Discount);
```

```
END sub_getPrice_QA_V1;
```

3.3 Starting the WebSphere MQ Adapter

The process for starting the adapter varies based on the operating system.

- To start the WebSphere MQ adapter on Unix:
  1. Change to the directory containing the start script.

     ```
     cd ORACLE_HOME/integration/interconnect/adapters/Application
     ```

  2. Type **start** and press **Enter**.
- To start the WebSphere MQ adapter from Services on Windows:
  1. Access the Services window from the Start menu.

     The Services window is displayed.
  2. Select the ***OracleHomeOracleASIntegrationInterConnectAdapter-Application*** service.
  3. Start the service based on the operating system.

     The WebSphere MQ adapter automatically starts the publishing engine, a tool for notifying foreign applications of additions, deletions, or updations to the native application.

Note: You can also start and stop the WebSphere MQ adapter using the IC Manager. Refer to *Oracle Application Server Integration InterConnect User's Guide* for more details.

3.3.1 Log File of WebSphere MQ Adapter

You can verify the start up status of the WebSphere MQ adapter by viewing the `log.xml` files. The files are located in the time-stamped subdirectory of the `log` directory of the WebSphere MQ adapter. Subdirectory names have the following form:

```
timestamp_in_milliseconds
```

The following is an example of the information about an WebSphere MQ adapter that started successfully:

```
The Adapter service is starting..
Registering your application (MQAPP)..
Initializing the Bridge oracle.oai.agent.adapter.mqseries.MQBridge..
Starting the Bridge oracle.oai.agent.adapter.mqseries.MQBridge..
Service started successfully.
```

3.4 Stopping the WebSphere MQ Adapter

The process for stopping the adapter varies based on the operating system.

- To stop the WebSphere MQ adapter on UNIX:
  1. Change to the directory containing the stop script.

     ```
     cd ORACLE_HOME/integration/interconnect/adapters/Application
     ```

2. Type **stop** and press **Enter**.

- To stop the WebSphere MQ adapter from Services on Windows.

  1. Access the Services window from the Start menu.
  2. Select the ***OracleHomeOracleASInterConnectAdapter-Application*** service.
  3. Stop the service based on the operating system.

You can verify the stop status of the WebSphere MQ adapter by viewing the `log.xml` files. These files are located in the time-stamped subdirectory of the `log` directory of the WebSphere MQ adapter.

A Frequently Asked Questions

This appendix provides answers to following frequently asked questions about the WebSphere MQ adapter.

- How do I know the WebSphere MQ adapter has started properly?
- The WebSphere MQ adapter did not start properly. What went wrong?
- My WebSphere MQ adapter is not starting. What could be the reason?
- Is it possible to edit the WebSphere MQ adapter configuration settings created during installation?
- When I change an element in iStudio, such as mappings, it seems like the WebSphere MQ adapter is using old information. What is happening?
- How do I secure my passwords?
- I am getting a JMS-nnnn error when the WebSphere MQ adapter is starting up. What is wrong?
- I am sending files with names such as MQ.FailedMsg.<message-id> in the directory where the adapter.ini file is located. What does this mean?
- Why am I getting a "oracle.oai.agent.adapter.sdk.Agent.createMessageObject(xml)" error in the log file?
- Why do I get the "Unable to load message catalog: mqji" error message when starting the WebSphere MQ adapter?

A.1 How do I know the WebSphere MQ adapter has started properly?

View the `log.xml` file located in the time-stamped subdirectory of the WebSphere MQ adapter log directory:

Platform	Directory
UNIX	`ORACLE_HOME/integration/interconnect/adapters/Application/log/timestamp_in_milliseconds`
Windows	`ORACLE_HOME\integration\interconnect\adapters\Application\log\timestamp_in_milliseconds`

If there are no exceptions, then the WebSphere MQ adapter has started properly.

A.2 The WebSphere MQ adapter did not start properly. What went wrong?

View the exceptions in the WebSphere MQ adapter log file (`log.xml`). The exceptions should provide information about what went wrong. It is possible that the WebSphere MQ adapter is unable to connect to the repository. Ensure the repository is started properly. The WebSphere MQ adapter will connect to the repository once it is started properly. You do not need to restart the Adapter.

See Also: *Oracle Application Server Integration InterConnect User's Guide* for instructions on starting the repository on UNIX and Windows

A.3 My WebSphere MQ adapter is not starting. What could be the reason?

One reason can be that Oracle Wallet does not contain the password information corresponding to your application name. For example, during installation you defined the application name as `myMQApp`. Later, you changed the application name in iStudio to `MQApp`. In such case, you need to specify the password corresponding to the new application name `MQApp` in the Oracle Wallet. You can create password by using the `oraclewallet` command.

See Also: Section A.6, "How do I secure my passwords?"

A.4 Is it possible to edit the WebSphere MQ adapter configuration settings created during installation?

Yes, edit the parameters in the `adapter.ini` file in the following directory:

Platform	**Directory**
UNIX	`ORACLE_HOME/integration/interconnect/adapters/Application/`
Windows	`ORACLE_HOME\integration\interconnect\adapters\Application\`

See Also: Chapter 2, "Installation and Configuration"

A.5 When I change an element in iStudio, such as mappings, it seems like the WebSphere MQ adapter is using old information. What is happening?

The WebSphere MQ adapter caches information from iStudio. The information is stored in the repository locally. If you change something in iStudio and want to view the change in the run time, then you need to stop the WebSphere MQ adapter, delete the WebSphere MQ adapter cache files, and restart the WebSphere MQ adapter.

The WebSphere MQ adapter has a persistence directory which is located in the WebSphere MQ adapter directory. Deleting this directory when the WebSphere MQ adapter has been stopped should make it obtain the new metadata from the repository when started.

A.6 How do I secure my passwords?

OracleAS Integration InterConnect uses Oracle Wallet Manager to maintain system passwords. When you install OracleAS Integration InterConnect, Oracle Wallet Manager is also installed and a password store is created. All passwords used by OracleAS Integration InterConnect components are stored in the password store. The password is stored in the Oracle Wallet in the following format:

```
ApplicationName/password
```

The `ApplicationName` is the name of the application, which is extracted from the `adapter.ini` file of the corresponding adapter. In the `adapter.ini` file, the `application` parameter specifies the `ApplicationName` to which this adapter connects. The password for the application is also retrieved from the `adapter.ini` file.

The number of entries is dependent on the type of adapter. For example, DB Adapter needs two entries whereas AQ Adapter needs only one entry. The following table lists the entries that will be created for each adapter:

Adapter	**Entry In Oracle Wallet**
AQ	*ApplicationName*/aq_bridge_password
HTTP	*ApplicationName*/http.sender.password
HTTP	*ApplicationName*/sender.wallet_password
SMTP	*ApplicationName*/smtp.receiver.password
MQ	*ApplicationName*/mq.default.password
FTP	*ApplicationName*/file.sender.password
FTP	*ApplicationName*/file.receiver.password
DB	*ApplicationName*/db_bridge_schema1_password
DB	*ApplicationName*/db_bridge_schema1_writer_password

You can create, update, and delete passwords using the `oraclewallet` command. When you run the command, it prompts you for the admin password.

You can use the following commands to manage your passwords:

- List all passwords in the store

  ```
  oraclewallet -listsecrets
  ```

- Create a password

  ```
  oraclewallet -createsecret passwordname
  ```

  For example, to create a password for the hub schema:

  ```
  oraclewallet -createsecret hub_password
  ```

- View a password

  ```
  oraclewallet -viewsecret passwordname
  ```

  For example, to view the password for the hub schema:

  ```
  oraclewallet -viewsecret hub_password
  ```

- Update a password

  ```
  oraclewallet -updatesecret passwordname
  ```

  For example, to update the password for the hub schema:

  ```
  oraclewallet -updatesecret hub_password
  ```

- Delete a password

  ```
  oraclewallet -deletesecret passwordname
  ```

  For example, to delete the password for the hub schema:

  ```
  oraclewallet -deletesecret hub_password
  ```

A.7 I am getting a JMS-nnnn error when the WebSphere MQ adapter is starting up. What is wrong?

Look up the error code in the *IBM WebSphere MQ for Java guide* Messages Appendix and correct any mistakes for the WebSphere MQ connection information in `adapter.ini`. The following lists some common error codes:

- `2009 MQRC_CONNECTION_BROKEN`: The connection to the queue manager has been lost. This can occur because the queue manager has ended. All previous handles are now invalid. As a result, the WebSphere MQ adapter should be restarted.
- `2030 MQRC_MSG_TOO_BIG_FOR_Q`: The message length is greater than the maximum for the queue. Increase `MaxMsgLength` for the queue (WebSphere MQ Administrator).
- `2031 MQRC_MSG_TOO_BIG_FOR_Q_MGR`: The message length is greater than the maximum allowed by the remote queue manager. This error also occurs if the message size is larger than the maximum message size allowed by a channel through which the message is to pass.
- `2035 MQRC_NOT_AUTHORIZED`: The user is not authorized to perform the operation attempted. Make sure the `mq.default.user` and `mq.default.password` parameters in `adapter.ini` are correct.

More error codes can be found at the following url: `http://www-4.ibm.com/software/ts/mqseries/library/manuals/csqfao/CSQFAO1P.HTM`.

A.8 I am sending files with names such as MQ.FailedMsg.<message-id> in the directory where the adapter.ini file is located. What does this mean?

The means that some outbound messages received from WebSphere MQ did not parse successfully with any of the registered D3L files. Either one or more D3L files should be corrected or the WebSphere MQ sending agent, which enqueued the message on the outbound queue, should correct the messages so they conform to one of the D3L files. If you configure the `mq.default.receiver.exception.uri` parameter in the `adapter.ini` file, then the 'failed' messages will be enqueued on the configured exception queue.

A.9 Why am I getting a "oracle.oai.agent.adapter.sdk.Agent.createMessageObject(xml)" error in the log file?

The complete text of the error message is "`MQMessageSender_run`: The following exception occurred while invoking `oracle.oai.agent.adapter.sdk.Agent.createMessageObject(xml).` If the Published Message Type in iStudio was XML, then try instead to use the Message Type Generic, setting the Object name to be the root element of the XML document."

The error message essentially also provides the solution to this problem.

A.10 Why do I get the "Unable to load message catalog: mqji" error message when starting the WebSphere MQ adapter?

This is a benign warning message from the WebSphere MQ Java layer which can be avoided by adding the `/opt/mqm/java/lib` directory to the Java CLASSPATH before starting the WebSphere MQ adapter (modifying the `start` script).

B Example of the adapter.ini File

This appendix shows a sample `adapter.ini` file for the WebSphere MQ adapter.

> **See Also:** Configuring the WebSphere MQ Adapter on page 2-7 for additional information on `adapter.ini` configuration parameters

The following code is an example of the FTP `adapter.ini` file.

```
#include <../../hub/hub.ini>

// *************
// ** Adapter **
// *************

// Application (as created in iStudio) corresponding to this Adapter.
application=myFtpApp

// Partition (as created in iStudio) corresponding to this Adapter.
partition=

// If you have multiple adapter instances for a given application with the
// given partition, each Adapter should have an instance number.

//instance_number=2

// Bridge class
bridge_class=oracle.oai.agent.adapter.technology.TechBridge

ota.type=D3L


// define the ftp sending endpoint
// For ftp, ota.send.endpoint=ftp://<host name>/<path name>
// For file, ota.send.endpoint=file://<host name>/<path name>
//
ota.send.endpoint= ftp://foo.s.com/private/ipdev1/test/d3l/inbound

// define the ftp receiving endpoint
// For ftp, ota.send.endpoint=ftp://<host name>/<path name>
// For file, ota.send.endpoint=file://<host name>/<path name>
//
ota.receive.endpoint=ftp://foo.s.com/private/ipdev1/test/d3l/inbound


//-------------------------------------
```

```
// ftp Sender initialization variables
//-----------------------------------------

// ftp user (mandatory if ftp is used)
// file.sender.user=ipdev1
file.sender.user=ipdev1


// ftp user password (mandatory if ftp is used)
//file.sender.password=ipdev1
file.sender.password=ipwelcome


// file type (ASCII or BINARY)
//file.sender.type=BINARY
file.sender.type=ASCII

// proxy host
//file.sender.proxy_host=

// proxy port
//file.sender.proxy_port=
//staging directory
//file.sender.staging_directory =/tmp

//sender customizer class
//file.sender.customizer_class = MySenderCustomizer

//-----------------------------------------
// ftp receiver initialization variables
//-----------------------------------------

// ftp user (mandatory if ftp is used)
//file.receiver.user=ipdev1
file.receiver.user=ipdev1

// ftp user password (mandatory if ftp is used)
//file.receiver.password=ipdev1
file.receiver.password=ipwelcome

// file type (ASCII or BINARY)
//file.receiver.type=BINARY
file.receiver.type=BINARY

// proxy host
//file.receiver.proxy_host=

// proxy port
//file.receiver.proxy_port=
//receiver customizer class
//file.receiver.customizer_class = MyReceiverCustomizer

// define where to put the
// file that cannot be processed
// properly.
//file.receiver.exception_dir=


// define how often to poll
// the message source (in milli seconds)
```

```
file.receiver.polling_interval=60000

// define maximum number of messages
// retrieved in each polling session
file.receiver.max_msgs_retrieved=30

// D3L initialization variables
ota.d3ls=person2.xml:person1.xml


// *************
// ** Agent ***
// *************

// Log level (0 = errors only, 1 = status and errors, 2 = trace, status and
errors).
agent_log_level=2

// Hub message selection information
agent_subscriber_name=myFTPApp
agent_message_selector=recipient_list like '%,myFTPApp,%'
// Only provide values for the next two parameters if you have multiple Adapter
instances for the given application with the given partition.

//agent_reply_subscriber_name=
//agent_reply_message_selector=

// Set this to false if you want to turn off all tracking of messages (if true,
messages which
have tracking fields set in iStudio will be tracked)
agent_tracking_enabled=true

// Set this to false if you want to turn off all throughput measurements
agent_throughput_measurement_enabled=true

// By default, adapters use an OAI specific DTD for all messages sent to the Hub
//because other OAI adapters will be picking up the messages from the Hub and know
// how to interpret them. This should be set to true if for every message, you
//would like to use the DTD imported for that message's Common View instead
//of the OAI DTD. This should only be set to true if an OAI Adapter
//is *NOT* receiving the messages from the Hub.

agent_use_custom_hub_dtd=false

// Sets the metadata caching algorithm. The possible choices are startup (cache
everything at
startup: this may take a while if there is a lot of metadata in your Repository),
demand (cach
e metadata as it is used) or none (no caching: this will slow down performance.)
agent_metadata_caching=demand

// Sets the DVM table caching algorithm. The possible choices are startup (cache
all DVM table
s at startup: this may take a while if there are a lot of tables in your
Repository), demand (
cache tables as they are used) or none (no caching: this will slow down
performance.)
agent_dvm_table_caching=demand

// Sets the lookup table caching algorithm. The possible choices are startup
```

```
(cache all lookup
 tables at startup: this may take a while if there are a lot of tables in your
Repository), de
mand (cache tables as they are used) or none (no caching: this will slow down
performance.)
agent_lookup_table_caching=demand

// If metadata caching, DVM table caching, or lookup table caching are turned on
//(startup or demand) then the Adapter caches metadata or DVM tables it retrieves
//from the Repository in a file cache. When you restart the Adapter,it will not
// have to get that metadata or DVM table from the Repository again because it is
// in the cache files.However, if you change some metadata or DVM table using
// iStudio and you want the Adapter to use those changes the next time it is
// started you can either delete the cache files or set this parameter to true
// before restarting.
agent_delete_file_cache_at_startup=false

// Max number of application data type information to cache
agent_max_ao_cache_size=200

// Max number of common data type information to cache
agent_max_co_cache_size=100

// Max number of message metadata to cache
agent_max_message_metadata_cache_size=200

// Max number of DVM tables to cache
agent_max_dvm_table_cache_size=200

// Max number of lookup tables to cache
agent_max_lookup_table_cache_size=200

// Internal Agent queue sizes
agent_max_queue_size=1000
agent_Persistence_queue_size=1000

// Persistence
agent_persistence_cleanup_interval=60000
agent_persistence_retry_interval=60000


 //////////////////
 // End Comments //
//////////////////
```

Index