Oracle® Application Server Integration InterConnect

Adapter for AQ Installation and User's Guide

10*g* Release 2 (10.1.2)

B14077-02

December 2005

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

B14077-02

Primary Author: Rima Dave

Contributing Author: Vimmika Dinesh

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 AQ Installation and User's Guide is intended for system administrators of OracleAS Integration InterConnect 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 on how to use Oracle Application Server Integration InterConnect (OracleAS Integration InterConnect) Adapter for Advanced Queuing (AQ adapter). It contains the following topics:

- AQ Adapter Overview
- AQ Adapter System Requirements

1.1 AQ Adapter Overview

The AQ adapter enables an Oracle Advanced Queuing application to be integrated with other applications that use OracleAS Integration InterConnect. The AQ adapter is useful in all Enterprise Application Integration (EAI) scenarios involving Oracle Database Advanced Queuing applications. EAI is the integration of applications and business processes within the same company.

1.2 AQ Adapter System Requirements

The following sections describe the AQ adapter system requirements:

- Hardware Requirements
- Software Requirements

1.2.1 Hardware Requirements

Table 1–1 lists the hardware requirements for installing the AQ adapter:

Table 1–1 Hardware Requirements

Hardware	Windows 2000	UNIX
Disk Space	400 MB	400 MB
Memory	512 MB	512 MB

1.2.2 Software Requirements

The following sections describe the AQ adapter software requirements:

- Operating System Requirements
- JRE Requirements
- Database Requirements

Operating System Requirements

Table 1–2 lists the operating system requirements for installing the AQ 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.

Database Requirements

The AQ adapter requires Oracle Database 8*i* and later versions.

Note: The AQ adapter can be installed on a remote computer. The Oracle Advanced Queuing Application does not need to be installed on the spoke computer.

2 Installation and Configuration

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

- Installing the AQ Adapter
- Installing Multiple AQ Adapters in the Same Oracle Home
- Configuring the AQ Adapter

2.1 Installing the AQ Adapter

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

This section describes the following topics:

- Preinstallation Tasks
- Installation Tasks

2.1.1 Preinstallation Tasks

Refer the following guides before installing the AQ adapter:

- *Oracle Application Server Installation Guide* for information about Oracle Universal Installer (OUI) startup.
- *Oracle Application Server InterConnect Installation Guide* for information on software, hardware, and system requirements for OracleAS Integration InterConnect.

2.1.2 Installation Tasks

To install the AQ adapter:

1. Select **OracleAS Integration InterConnect Adapter for AQ 10.1.2.0.2** in the Available Product Components screen of the OracleAS Integration InterConnect installation wizard.
2. Click **Next**. The Set Oracle Wallet Password screen is displayed.
3. Enter and confirm the password on the screen, which will be used to manage OracleAS Integration InterConnect installation.
4. Click **Next**.
   - Go to step 5 if installing the AQ 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 6 if installing the AQ 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.

5. 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 System Identifier (SID) for the hub database.
   - Password: The user password for the hub database user.
6. Click **Next**. The Specify AQ Adapter Name screen is displayed.
7. Enter the application name. Blank spaces are not permitted. The default value is `myAQApp`.

   Note: You can change the application name in iStudio after installation. In such 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, "The AQ adapter is not starting. What could be the reason?"
   - Section A.11, "How do I secure my passwords?"
8. Click **Next**. The Specify Spoke Application Database Connection screen is displayed. This screen configures the information for the spoke application database. Enter information in the following fields:
   - Host Name: The name of the computer where the spoke application database is installed.
   - Port Number: The TNS listener port for the spoke application database.
   - Database SID: The SID for the spoke application database.

   The information on this screen is for Advanced Queues, from which the adapter will deliver or receive messages.
9. Click **Next**. The Spoke Application Database AQ Username screen is displayed. Enter information in the following fields:
   - User Name: The name the AQ adapter uses to connect to the database.
   - Password: The password for the user name.
   - Consumer Name: The consumer name used by the application that writes to the queue. The consumer name indicates that OracleAS Integration InterConnect should pick up a message. Alternatively, the AQ adapter may have a subscriber configured to which the AQ adapter corresponds.

     Leave this field blank if the queues that the AQ adapter will connect to on the application database side are single consumer queues. However, if any of the queues are multiconsumer queues, then specify a consumer name.

     Use one of the following methods to determine the consumer name to use:

  - If the code that will write a message to the queue is already available, then look at the code or the documentation that comes with it to determine the consumer name.
  - If the code that will write a message to the queue is not written, then enter a string as the consumer name. When the code is built, ensure that the consumer names match. Alternatively, if the queue has a subscriber configured, then use the database's Advanced Queuing APIs to find the name.

10. Click **Next**. The Summary screen is displayed.
11. Click **Install** to install the AQ adapter and other selected components. The AQ adapter is installed in the following directory:

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

`Application` is the value specified in Step 7.

12. Click **Exit** on the Installation screen to exit the AQ adapter installation.

2.2 Installing Multiple AQ Adapters in the Same Oracle Home

To install multiple instances of the AQ 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 AQ adapter with name `myAQApp` installed on a computer. To install another instance of the AQ adapter with name `myAQApp1` in the same Oracle home, use the following command:

```
copyAdapter myAQApp myAQApp1
```

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 AQ adapter `myAQApp1` by using the `copyAdapter` script. To manage the `myAQApp1` 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="myAQApp1" module-id="adapter" working-dir="$ORACLE_
HOME/integration/interconnect/adapters/myAQApp1" status="enabled">
       <start timeout="600" retry="2"/>
       <stop timeout="120"/>
       <port id="icadapter_dmsport_range" range="15701-15800"/>
       <process-set id="myAQApp1" 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="myAQApp" module-id="adapter" working-dir="$ORACLE
_HOME/integration/interconnect/adapters/myAQApp" status="enabled">
       <start timeout="600" retry="2"/>
       <stop timeout="120"/>
       <port id="icadapter_dmsport_range" range="15701-15800"/>
       <process-set id="myAQApp" 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="myAQApp1" module-id="adapter" working-dir="$ORACLE
_HOME/integration/interconnect/adapters/myAQApp1" status="enabled">
       <start timeout="600" retry="2"/>
       <stop timeout="120"/>
       <port id="icadapter_dmsport_range" range="15701-15800"/>
       <process-set id="myAQApp1" 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 `myAQApp1` adapter by using the following command

   ```
   opmnctl startproc ias-component="InterConnect" process-type="myAQApp1"
   ```

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.11, "How do I secure my passwords?"in Appendix A, "Frequently Asked Questions"

2.3 Configuring the AQ Adapter

You can configure the AQ adapter 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 AQ Adapter Directory

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

Table 2–2 describes the various executable files of the AQ 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 AQ adapter configuration files.

Table 2–3 Configuration Files

File	Description
`adapter.ini` (UNIX)	Contains all the initialization parameters, which the adapter reads at startup.
`adapter.ini` (Windows)	Contains all the initialization parameters, which the adapter reads at startup.

Table 2–4 describes the directories used by the AQ adapter.

Table 2–4 Directories

File	Description
`logs`	The adapter activity is logged in subdirectories of the `logs` directory. Each time the adapter is run, a new subdirectory is created for the `log.xml` log file.
`persistence`	The messages are persisted in this directory. Do not edit this directory or its files.

2.3.1 Using the Application Parameter

Adapters do not have integration logic. The AQ adapter has a generic transformation engine that processes 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 allows the adapter to retrieve only the relevant metadata from the repository. The application parameter must match the corresponding application that will be defined in iStudio, under the Applications folder.

If you use pre-packaged 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.

See Also: Step 6 on page 2-2

2.3.2 AQ Adapter Ini File Settings

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

- hub.ini File
- adapter.ini File

2.3.2.1 hub.ini File

The AQ 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 an example for each parameter.

Table 2–5 hub.ini Parameters

Parameters	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). There is no default value.	`hub_username=myhub`
`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

For a hub installed on an 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 nod	`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 File

The AQ adapter connects to the spoke application using parameters from the `adapter.ini` file. Table 2–7 lists the parameters, their description, and an example for 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 start up. 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 start up. 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`
`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 start up. 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`

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

Parameter	Description	Example
`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 the 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,%`
`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`

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

Parameter	Description	Example
`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 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 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,%'`
`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=aqapp and instance_number=2, then agent_reply_subscriber_name=aqapp2`
`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=aqapp`
`agent_throughput_measurement_enabled`	Specifies if the throughput measurement is enabled. Set this parameter to `true` to activate throughput measurements. Possible value: `true` or `false` 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. Possible value: `true` or `false` Default value: `true`	`agent_tracking_enabled=true`

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

Parameter	Description	Example
`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=aqapp`
`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 is used to explicitly specify the encoding of the published message. This parameter is ignored when the encoding already exists in the subscribed message.	`encoding=Shift_JIS`
`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:\`*`ORACLE_HOME`*`\integration\interconnect\adapters\aqapp\`
`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`

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

Parameter	Description	Example

`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 in iStudio. Possible value: An alphanumeric string Default value: None	`partition=germany`
`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`

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

Parameter	Description	Example
`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: The 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: The valid JVM heap size Default value: `536870912` This parameter is only for Microsoft Windows.	`service_min_heap_size=536870912`
`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: The 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: A valid JVM argument
	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 the 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`
`s`	Second in minute	`55`
`S`	Millisecond	`978`

Advanced Queuing Adapter-Specific Parameters

Table 2–9 lists the parameters specific to the AQ adapter.

Table 2–9 AQ Adapter-Specific Parameters

Parameter	Description	Example
`aq_bridge_consumer_name`	If all the queues that this adapter will connect to on the application database side are single consumer queues, then this can be left blank. If, however, any of the queues is a multiconsumer queue, then specify a consumer name. Possible value: `aq_bridge_username` Default value: None **Note**: If you are integrating B2B with OracleAS Integration InterConnect, then set the parameter `aq_bridge_consumer_name` to `b2b_user`. Also set the AQ adapter to listen on `IP_IN_QUEUE`. For example, aq_bridge_consumer_name=`b2b_user`.	`aq_bridge_consumer_name=aquser`
`aq_bridge_host`	Name of the computer hosting the database instance specified by `aq_bridge_instance`. Default value: None	`aq_bridge_host=mpscott-pc`
`aq_bridge_instance`	The SID of the database instance. Default value: None	`aq_bridge_instance=orcl`

Table 2–9 (Cont.) AQ Adapter-Specific Parameters

Parameter	Description	Example
`aq_bridge_owner`	The owner of the advanced queue. Possible value: `aq_bridge_username` Default value: None	`aq_bridge_owner=aquser`
`aq_bridge_password`	The password corresponding to the `aq_bridge_username` parameter. Possible value: `aquser` Default value: None **Note**: All passwords are stored in Oracle Wallet. Refer to Section A.11, "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.	`aq_bridge_password=welcome`
`aq_bridge_port`	The port where the TNS listener is running for the database instance specified by `aq_bridge_instance`. Possible value: A TNS listener number Default value: None	`aq_bridge_port=1521`
`aq_bridge_thin_jdbc`	This indicates whether to use thin JDBC when talking to the database. Default value: `true`	`aq_bridge_thin_jdbc=true`
`aq_bridge_username`	The schema user name that the bridge should connect to which dequeus or enqueues messages from a queue in order to publish or subscribe to events defined using iStudio. Possible value: `aquser` Default value: None	`aq_bridge_username=aquser`
`bridge_class`	Specifies the entry class for the AQ adapter. Once set, the value cannot be modified. Default value: None	`bridge_class=oracle.oai.agent.adapter.aq.XMLAQBridge`

Oracle Real Application Clusters adapter.ini Parameters for the Advanced Queuing Adapter

When the AQ adapter is servicing a Real Application Clusters database as the spoke database, parameters listed in Table 2–10 represent information on connection and configuration.

Table 2–10 Real Application Clusters adapter.ini Parameters

Parameter	Description	Example
`aq_bridge_host`*x*	Indicates the host for node *x*.	`aq_bridge_host2=dsunram13`
`aq_bridge_instance`*x*	Indicates the instance on node *x*.	`aq_bridge_instance2=orcl2`
`aq_bridge_num_nodes`	Indicates the number of nodes in a cluster.	`aq_bridge_num_nodes=4`
`aq_bridge_port`*x*	Indicates the port for node *x*.	`aq_bridge_port2=1421`

3

Design-Time and Run-Time Concepts

This chapter describes the design-time and run-time concepts of the AQ adapter. It contains the following topics:

- Section 3.1, "AQ Adapter Design-Time Concepts"
- Section 3.2, "Designing with iStudio"
- Section 3.3, "AQ Adapter Run-Time Concepts"
- Section 3.4, "Starting the AQ Adapter"
- Section 3.5, "Stopping the AQ Adapter"

3.1 AQ Adapter Design-Time Concepts

The following topics describe the iStudio concepts pertinent to the AQ adapter. The AQ adapter can handle the following payload types:

- RAW Payload with XML data
- Oracle Object Payload with and without XML Data

3.1.1 RAW Payload with XML data

You can use iStudio to import a Document Type Definition (DTD) or XML Schema Definition(XSD) and configure an application where the corresponding message can be picked up or placed by the adapter. If the queue has been configured for RAW payload, then the message payload is plain XML.

See Also: Chapter 4, "Sample Use Cases"

3.1.2 Oracle Object Payload with and without XML Data

In addition to RAW payloads, the AQ adapter supports Oracle Object Types. The AQ adapter provides complete flexibility to import the Advanced Queue's Oracle Object Type payload. Thus, the attributes associated with objects within this Oracle Object Type can be of different XML types.

For example, assume that you want to send two objects, `Customer` and `PurchaseOrder`, as part of one OracleAS Integration InterConnectmessage. The corresponding DTDs are `customer.dtd` and `purchaseOrder.dtd`. When an Oracle Advanced Queue is *inQueue*, it contains an Oracle Object Type payload (`Customer CLOB`, `CreationDATE`, and `PurchaseOrder BLOB`). In this example, the application is enqueuing an Oracle Object containing Customer XML adhering to `customer.dtd`, a creation date, and a Purchase Order XML adhering to `PurchaseOrder.dtd`.

The following steps describe the tasks performed in iStudio to complete the example:

1. Create iStudio data types and import the corresponding XML DTDs.

   Note: This issue only affects users using queues with an Oracle Object Type payload.

   For example, create an application datatype called `DTDs` and then select Import from XML to import `customer.dtd`. Import `PurchaseOrder.dtd` in the same way. Select **Reload** from the **File** menu, then select the current project.

   Use the **Import From the Database** option when creating published events, subscribed events, invoked procedures, or implemented procedures.

   Note: Log in as the system user when importing DTDs on the Define Application View dialog.

   The three corresponding OracleAS Integration InterConnect attributes, `Customer String`, `CreationDate Date`, and `PurchaseOrder String` are created in iStudio.

2. Change the data type of the `Customer` attribute from string to the attribute created when `customer.dtd` was imported. Similarly, change the data type for the `PurchaseOrder` attribute to correspond to the one created using `PurchaseOrder.dtd`.

3.2 Designing with iStudio

The following steps describe how to create metadata using iStudio. To create metadata in iStudio, you should be familiar with the general process of creating metadata.

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

3.2.1 Importing from XML

1. Select **Import XML** on the Publish or Subscribe Wizard. The File dialog box is displayed.
2. Select the DTD file. A list of all nodes is displayed.
3. Select the root element.

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

The following are salient points when working with AQ adapter:

- Specify the Message Type as **AQ** for Advanced Queue applications.
- Common view: Create by importing from XML and specifying the DTD file.
- Application view:
  - Raw Payload: Import from XML.
  - Object Payload: Import from the database by selecting the queue payload.

- Event map usage: Event maps need to be used only if two or more events published by a particular application have the same application view structure.
- Follow these steps to specify the application queues:
  1. Expand the **Applications** node on the deploy tab in iStudio.
  2. Expand applicationName.
  3. Expand the Routing node.
  4. Right-click **Application Queues**, and click **Edit**.
  5. Enter the Application Queue Names for the AQ adapter to subscribe and publish messages to.
  6. Click **OK**.

3.2.2 Returned In Arguments

Returned In Arguments are used only when invoking procedures. Returned In Arguments propagate IN/OUT attributes contained in both request and reply messages. Without this feature, these IN/OUT attributes would have to exist in both the common view and the application view of the implementor and mappings would need to exist to copy these attributes between the views.

You can use one of these Returned In Arguments to correlate the reply with an asynchronous request.

For example, assume a Customer object exists which looks like the following in the application view:

```
Customer
  Name
  ID
  Contact
    Address
      City
      State
      Zip
    Phone
      AreaCode
      PhoneNumber
```

This Customer object is to be sent as part of a `CreateCustomer` message. If `ID` should be both in the request and the reply, then it should be an `IN/OUT` parameter. Click Returned In Args in the Invoke Wizard and select ID in the Please Select In Arguments and the Please Select Out Arguments dialogs.

3.3 AQ Adapter Run-Time Concepts

This section describes the run-time concepts of the AQ adapter. It contains the following topics:

- Advanced Queuing Sender
- Advanced Queuing Receiver

3.3.1 Advanced Queuing Sender

The AQ adapter consists of the bridge and the run-time agent. The bridge is constantly polling the queue chosen for publishing messages in the `aq_bridge_username` schema as specified in the `adapter.ini` file. A new message in this queue indicates a new outbound OracleAS Integration InterConnect message waiting to be sent by the adapter. The adapter then picks up the message, builds the corresponding OracleAS Integration InterConnect message, stores it, transforms it to the common view, and routes it to the hub. From the hub, the message is routed to the corresponding subscriber based on configuration done using iStudio, which could be content-based or subscription-based.

The application and the AQ adapter communicate using the publishing and invoking queues residing in the `aq_bridge_username` parameter for outbound messages and by subscribing and implementing queues for inbound messages. Thus, if the AQ adapter is down while the application is publishing OracleAS Integration InterConnect messages, these messages are held in the queues and will be picked up in the order they were enqueued by the AQ adapter once it is up and running. If there are messages in the queues that should no longer be published, then dequeue them manually.

3.3.2 Advanced Queuing Receiver

On the subscribing or receiving side, the AQ adapter receives the message from the hub, transforms it from common view to application view, and passes it to the bridge, which enqueues the message to the subscribe queue configured on the Deploy tab of iStudio. The application then picks this message from this queue. If the AQ adapter is an implementor instead of a subscriber, then the correlation fields are used to correlate between the request enqueued by the adapter and the reply enqueued by the application in the reply queue.

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.

3.4 Starting the AQ Adapter

Based on the operating system, the process for starting the adapter varies.

- To start the AQ 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 AQ adapter from Services on Windows:
  1. Access the Services window from the Start menu.
  2. Select the ***OracleHomeOracleASInterConnectAdapter-Application*** service.
  3. Start the service based on the operating system.

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

3.4.1 Log File of AQ Adapter Instance

You can verify the startup status of the AQ adapter by viewing the `log.xml` files. The files are located in the time-stamped subdirectory of the `log` directory in the AQ adapter directory. Subdirectory names take the following form:

```
timestamp_in_milliseconds
```

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

```
The Adapter service is starting..
Registering your application (AQAPP)..
Initializing the Bridge oracle.oai.agent.adapter.aq.XMLAQBridge..
AQ Adapter: created a reader for queue xml_q1.
Starting the Bridge oracle.oai.agent.adapter.aq.XMLAQBridge..
Service started successfully.
```

3.5 Stopping the AQ Adapter

Based on the operating system, the process for stopping the adapter varies.

- To stop the AQ 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 AQ 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 AQ adapter by viewing the `log.xml` files. These files are located in the time-stamped subdirectory of the `log` directory of the AQ adapter.

4 Sample Use Cases

This chapter describes the sample use cases for the AQ adapter.

4.1 Case One: Publish and Subscribe with RAW Payload

In this case, the AQ adapter publishes and subscribes using RAW Payload. This use case uses one installation of an AQ adapter that publishes a message and subscribes to the same event. Thus, the log files will indicate that an outbound message is followed by an inbound message.

4.1.1 Design Time Steps

This section describes the steps for publishing and subscribing a message containing RAW payload. The event created is `createCustomer` where the payload corresponds to the `customer.dtd` file. For outbound messages, the application queue is `xml_raw_q1`. For inbound messages, the application queue is `xml_raw_q2`.

1. Create a business object in iStudio. In the Create Business Object dialog box, enter `customer` in the Business Object Name field.
2. Create the event in iStudio. In the Create Event dialog box, complete the following:
   a. Select **customer** from Business Object.
   b. Enter `createCustomer` in the Event Name field.
   c. Click **Import** and select **XML** to import the `customer.dtd` file.
   d. Select **customer** as the root element.
3. Create an application in iStudio. In the Create Application dialog box, enter `aqapp` in the Application Name field.
4. Create a Published Event using the Publish Event Wizard in iStudio:
   a. In the Select an Event dialog box:
      * Select **aqapp** for the Application.
      * Select **AQ** as the Message Type.
      * Expand the list in the Select an Event box and select **createCustomer**.
   b. On the Define Application View dialog box:
      * Click **Import** and select **XML** to import the `customer.dtd` file.

  * Select **customer** as the root element.
  * Click **Tracking Fields** and select the **customer.id** tracking fields.

c. Define an ObjectCopy transformation from the application view customer to the common view customer on the Define Mappings dialog box.

d. Click **Finish**.

5. Create a Subscribed Event using the Subscribe Wizard in iStudio:

   a. On the Select an Event dialog box:

      * Select **aqapp** for the Application.
      * Select **AQ** as the Message Type.
      * Expand the list in the Select an Event box and select **createCustomer**.

   b. On the Define Application View dialog box:

      * Click **Import** and select **XML** on the Define Application View dialog box to import the `customer.dtd` file.
      * Select **customer** as the root element.

   c. Define a mapping so that ObjectCopy is completed from the application view customer to the common view customer on the Define Mappings dialog box.

   d. Click **Finish**.

6. Set up application queues in iStudio from the Deploy Navigation tab:

   a. Select **aqapp**, and then select **Application Queues**.

   b. Publish: createCustomer, Queue: `xml_raw_q1`

   c. Subscribe: createCustomer, Queue: `xml_raw_q2`

4.1.2 Run-Time Steps

Run the following script to create the application user:

```
connect system/manager
create user aquser identified by manager;
grant connect, resource to aquser;
grant aq_user_role, aq_administrator_role to aquser;
```

The following steps describe the runtime procedures necessary to publish and subscribe a message containing RAW XML payload.

1. Create the database user `aquser` by running the `create_user.sql` script.
2. Log in as `aquser/manager`.
3. Create Advanced Queues by executing the `CreateAQ.sql` script.

   See Also: "Design Time Steps" on page 4-4

4. Enqueue an XML message by executing the `EnqCust.sql` script.
5. Set the `agent_log_level` parameter to 2 in the `adapter.ini` file.
6. Delete the persistence directory and start the adapter.
7. Verify that the message has been published, subscribed to, and delivered to `xml_raw_q2` by viewing the log files in the log directory of the AQ adapter.

4.1.3 Related Files

The following files are related to the steps in case one.

- DTD to be imported:

```
customer.dtd
<!ELEMENT customer (id,name,address*)>
<!ELEMENT address (city, state)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT id (#PCDATA)>
```

- Script to create the RAW Queues `xml_raw_q1`, `xml_raw_q2`:

```
CreateAQ.sql
EXECUTE dbms_aqadm.create_queue_table (queue_table => 'RawMsgs_qtab', queue_
payload_type => 'RAW', multiple_consumers => FALSE);
EXECUTE dbms_aqadm.create_queue (queue_name => 'xml_raw_q1', queue_table =>
'RawMsgs_qtab');
EXECUTE dbms_aqadm.start_queue (queue_name => 'xml_raw_q1');
EXECUTE dbms_aqadm.create_queue (queue_name => 'xml_raw_q2', queue_table =>
'RawMsgs_qtab');
EXECUTE dbms_aqadm.start_queue (queue_name => 'xml_raw_q2');
```

- Script to Enqueue a `createCustomer` event on `xml_raw_q1 queue`:

```
EnqCust.sql
DECLARE
  enqueue_options     dbms_aq.enqueue_options_t;
  message_properties  dbms_aq.message_properties_t;
  msgid               RAW(16);
  payload             RAW(5000);
BEGIN
  payload := utl_raw.cast_to_raw('<?xml version="1.0" standalone="no"?>
      <customer><id>10</id>
      <name>Herb Stiel</name>
      <address>
  <city>SanMateo</city>
        <state>California</state>
     </address>
    </customer>');
  dbms_aq.enqueue(queue_name         => 'xml_raw_q1',
          enqueue_options    => enqueue_options,
          message_properties => message_properties,
          payload            => payload,
          msgid              => msgid);
  COMMIT;
END;
/
```

- The create_user.sql script:

```
CONNECT SYSTEM/MANAGER
CREATE USER aquser IDENTIFIED BY manager;
GRANT CONNECT, RESOURCE TO aquser;
GRANT AQ_USER_ROLE, AQ_ADMINISTRATOR_ROLE TO aquser;
```

4.2 Case Two: Invoke and Implement with Oracle Object Type Payload

In this case, the AQ adapter invokes and implements using Oracle Object Type Payload. The following case uses one installation of an AQ adapter that sends a request, receives the request, sends back a reply, and receives the reply. Thus, the log files will indicate four different message entries. The application used is `aqapp`.

4.2.1 Design Time Steps

This section describes the steps for invoking and implementing a procedure.

1. Log in as the `aquser/manager`.
2. Create the `addr` and `cust` Oracle Object Types by executing the `CreateADT.sql` script.
3. Create the necessary queues by executing the `CreateADTQueue.sql` script.
4. Create a Business Object in iStudio. In the Create Business Object dialog box, enter `customer` in the **Business Object Name** field.
5. Create a Procedure in iStudio. In the Create Procedure dialog box, complete the following:
   a. Select **customer** as the Business Object.
   b. Enter `updateCustomer` in the Procedure Name field.
   c. Click **Import** and select **XML** to import the `customer.dtd` file.
   d. Select the **IN/OUT arguments** option.
6. In the Create Data Type dialog box, complete the following:
   a. Enter `DTDs` in the Common Data Type Name field.
   b. Click **Import** and select **XML** to import the `customer.dtd` file.
   c. Select **customer** as the root element.
7. Reload the project.
8. Using the Invoke Wizard, complete the following:
   a. On the Select a Procedure dialog box:
      * Select **aqapp** for the Application.
      * Select **AQ** for the Message Type.
      * Expand the list in the **Select a Procedure** box and select **updateCustomer**.
   b. On the Define Application View dialog box:
      * Click **Import** and select **Database**.
      * Log in as the system user on the Database Login dialog box.
      * On the Oracle Database Browser dialog box, expand the **AQUSER** list and select **AQUSER.MY_QUEUE_TYPE**.
      * Click **Done**.
   c. Create the following mapping on the Define Mapping IN Arguments dialog box:
      * For aqapp view to the common view: `updateCustomer:IN - ObjectCopy - updateCustomer:IN`

   d. Create the following mapping on the Define Mapping OUT Arguments dialog box:

      * For common view to application view: `updateCustomer:OUT - ObjectCopy - updateCustomer:OUT`

9. Repeat step 8 to create a new implemented procedure using the Implement Wizard.
10. Set up application queues in iStudio from the Deploy Navigation tab:
    a. Select **aqapp**, then select **Application Queues**.
    b. Send Request: updateCustomer, Queue: `xml_q1`
    c. Receive Request: updateCustomer, Queue: `xml_q2`
    d. Send Reply: updateCustomer, Queue: xml_q2
    e. Receive Reply: updateCustomer, Queue: `xml_q3`

4.2.2 Run-Time Steps

The following steps describe the runtime procedures that implement and invoke a procedure with Oracle Object Type payload.

1. Execute the `EnqueueADT.sql` script to enqueue an XML message.

   See Also: "Design Time Steps" on page 4-4

2. Set the `agent_log_level` parameter to 2 in the `adapter.ini` file.
3. Delete the persistence directory and start the adapter.
4. Verify the following by viewing the adapter log files:
   - Request was dequeued from `xml_q1` and enqueued to the hub queue `oai_hub_queue`.
   - Request was dequeued from `oai_hub_queue` and enqueued to `xml_q2`.
   - Reply was dequeued from `xml_q2` and enqueued to the hub queue `oai_hub_queue`.
   - Reply was dequeued from `oai_hub_queue` and enqueued to `xml_q3`.

4.2.3 Related Files

The following files are related to the runtime steps in case two.

- `CreateADT.sql`

```
CREATE TYPE my_queue_type as object(id number, payload varchar2(1000));
/
```

- `CreateADTQueue.sql`

```
EXECUTE dbms_aqadm.create_queue_table (queue_table => 'ADTMsgs_qtab', queue_
payload_type => 'my_queue_type', multiple_consumers =>
FALSE);

EXECUTE dbms_aqadm.create_queue (queue_name => 'xml_q1', queue_table =>
'ADTMsgs_qtab');
EXECUTE dbms_aqadm.start_queue (queue_name => 'xml_q1');
EXECUTE dbms_aqadm.create_queue (queue_name => 'xml_q2', queue_table =>
'ADTMsgs_qtab');
EXECUTE dbms_aqadm.start_queue (queue_name => 'xml_q2');

EXECUTE dbms_aqadm.create_queue (queue_name => 'xml_q3', queue_table =>
'ADTMsgs_qtab');
EXECUTE dbms_aqadm.start_queue (queue_name => 'xml_q3');
```

- `EnqueueADT.sql`

```
DECLARE
      enqueue_options     dbms_aq.enqueue_options_t;
      message_properties  dbms_aq.message_properties_t;
      msgid               RAW(16);
      payload             cust;
BEGIN
      payload := my_queue_type(123,
                  '<customer><id>10</id>
                   <name>Herb Stiel</name>
                   <address>
                      <city>SanMateo</city>
                      <state>California</state>
                   </address>
                   </customer>');
dbms_aq.enqueue(queue_name         => 'xml_q1',
                     enqueue_options    => enqueue_options,
                     message_properties => message_properties,
                     payload            => payload,
                     msgid              => msgid);
      COMMIT;
END;
/
```

A Frequently Asked Questions

This chapter provides answers to frequently asked questions about the AQ adapter:

- How do I know that the AQ adapter has started properly?
- The AQ adapter did not start properly. What is wrong?
- The AQ adapter is not starting. What could be the reason?
- Why is the AQ adapter using old information after I changed information in iStudio?
- Which databases are referred to during installation?
- What consumer name one should provide during installation?
- The B2B is unable to pick up messages from InterConnect. The messages reach the IP_OUT_QUEUE, but stay there because B2B process them. What's wrong?
- How can I edit configuration settings after installation?
- How can I deliver a message to a specific partition of the publishing adapter?
- How do I handle ANY tags in DTDs imported into iStudio?
- How do I secure my passwords?

A.1 How do I know that the AQ adapter has started properly?

View the `log.xml` file located in the time-stamped subdirectory of the AQ 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`*

where *`Application`* is the value you defined in Step 6 on page 2-2, and *`timestamp_in_milliseconds`* is the directory. If no exceptions are listed, then the adapter has started properly.

A.2 The AQ adapter did not start properly. What is wrong?

View the exceptions in the AQ adapter log file (`log.xml`). The exceptions should provide some idea about what went wrong. It is possible that the AQ adapter is unable to connect to the repository. Ensure that the repository is started properly. Once the repository is started properly, the AQ adapter will connect to it. 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 The AQ 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 `myAQApp`. Later, you changed the application name in iStudio to `AQApp`. In such case, you need to specify the password corresponding to the new application name `AQApp` in the Oracle Wallet. You can create password by using the `oraclewallet` command.

See Also: Section A.11, "How do I secure my passwords?"

A.4 Why is the AQ adapter using old information after I changed information in iStudio?

The AQ adapter caches the information from iStudio that is stored locally in the repository for better performance in a production environment.

If you change something in iStudio and want to view it in the runtime environment, then stop the AQ adapter, delete the cache files, and restart the adapter.

Each adapter has a persistence directory located in the adapter's directory. Deleting this directory when adapter has been stopped allows the adapter to obtain the new metadata from the repository when started.

A.5 Which databases are referred to during installation?

The database referred during installation are those on the application side from which the adapter will either put or get messages from Advanced Queuing.

A.6 What consumer name one should provide during installation?

If all the queues that the AQ adapter connects to on the application side are single consumer queues, then leave this blank. However, if any one of the queues is a multiconsumer queue, then specify a consumer name.

The application that writes to the AQ adapter uses a consumer name to indicate to OracleAS Integration InterConnect to pick up this message. Use one of the following methods to determine the consumer name to use:

- If the piece of code that writes the message to the AQ adapter is already available, then look at that code or the documentation that comes with it to find the consumer name.

- If the piece of code that writes the message to the AQ adapter is not available, then enter any string as the consumer name. When that piece of code is built, ensure that the consumer names match.

A.7 The B2B is unable to pick up messages from InterConnect. The messages reach the IP_OUT_QUEUE, but stay there because B2B process them. What's wrong?

B2B listens on the `IP_OUT_QUEUE` as `b2bUser`, whereas InterConnect does not enqueue the message for any specific user. As a result, B2B does not pick up the messages.

Use the 'AddHeader' transformation in the subscribing adapter transformation mapping. The field should be 'aq_recipients' and the value should consist of comma-delimited consumer names.

A.8 How can I edit configuration settings after installation?

Edit the parameters in the following file:

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

The following table lists the parameters and their corresponding questions in the installation:

Parameter	Parameter Information
`aq_bridge_consumer_name`	The consumer name
`aq_bridge_host`	Host
`aq_bridge_instance`	The database SID
`aq_bridge_owner`	The Advanced Queuing owner. Enter the value if your Advanced Queuing adapter is installed under a different user than `aq_bridge_username`.
`aq_bridge_port`	The TNS listener port
`aq_bridge_thinjdbc`	Use a THIN JDBC driver if true, otherwise use OCI8 JDBC driver.
`aq_bridge_username`	User name

A.9 How can I deliver a message to a specific partition of the publishing adapter?

Specify the partition name before publishing a message to the queue specified in iStudio, for the publish event corresponding to this message. You can do this by using a trigger or Java AQ API. When using Java AQ API , specify the partition name for the

AQ Agent. When using a trigger, specify the specify the application name followed by the partition name as value of variable `recip_agent`. For example, application `AQAPP` has two partitions `PAR1` and `PAR2`. An AQ adapter is publishing an event `Create_Customer` to the hub queue. To enable the `PAR2` partition to receive message from the AQ and publish it to the hub, create the following trigger for a database table and specify the application name followed by the partition name as value of variable `recip_agent`:

```
CREATE OR REPLACE TRIGGER AQAPP.ENQUEUE_NEW_CUST
 BEFORE INSERT
 ON     AQAPP.CUSTOMER_TABLE
 FOR    EACH ROW
DECLARE
 qname VARCHAR2(20);
 enqueue_options DBMS_AQ.ENQUEUE_OPTIONS_T;
 message_properties DBMS_AQ.MESSAGE_PROPERTIES_T;
 msgid RAW(16);
 recip_agent SYS.AQ$_AGENT;
 raw_payload RAW(32767);
 payload VARCHAR2(256);
 recipient_list DBMS_AQ.AQ$_RECIPIENT_LIST_T;
BEGIN
 qname   := 'OUTBOUND_QUEUE';
 payload := '<?xml version="1.0" standalone="no"?>' ||
            '<Customer>'||
               '<id>' || :new.Id || '</id>' ||
               '<action>' || :new.Action || '</action>' ||
               '<name>' || :new.Name   || '</name>'   ||
               '<addr>'||
                    '<street>' || :new.Street || '</street>' ||
                    '<city>'   || :new.City   || '</city>'   ||
                    '<state>'  || :new.State  || '</state>'  ||
                    '<zip>'    || :new.Zip    || '</zip>'    ||
               '</addr>'  ||
            '</Customer>';
recip_agent := SYS.AQ$_AGENT('AQAPPPAR2',NULL,NULL);
--AQAPP is application name and PAR2 is partition name
recipient_list(1) := recip_agent;
message_properties.recipient_list := recipient_list;
raw_payload := UTL_RAW.CAST_TO_RAW( payload );
DBMS_AQ.ENQUEUE( queue_name => qname , enqueue_options => enqueue_options,message_
properties => message_properties, payload => raw_payload, msgid => msgid );
END;
```

A.10 How do I handle ANY tags in DTDs imported into iStudio?

`ANY` tags in an XML DTD allow unstructured data to be used in XML. OracleAS Integration InterConnect, however, must know about the structure of that data (using a DTD) if that data is to be used in mappings.

There are two methods for to know about the structure of the data:

1. The simplest method is to modify the DTD being imported into iStudio and replace the `ANY` tag with structured data. When modifying the DTD, only a copy of the DTD being imported into iStudio is modified, not the published version of the DTD. For example, if the `USERAREA ANY` tag is edited before importing the DTD into iStudio, only a copy is changed and the published OAG definition, which other people who download the OAG DTDs would use, is not changed.

This approach also supports using PCDATA for an ANY tag.

For example, consider the following customer.dtd:

```
<!ELEMENT customer (name, phone, address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT address ANY)>
```

This customer.dtd can be changed to the following:

```
<!ELEMENT customer (name, phone, address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT customer (name, phone, address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
```

This is dependent on what the XML will conform to at runtime. If the XML will use the ANY tag in different ways at runtime, a union can be used. For example, if address has street, city, and state only for some instances and for other instances has zip only, then a standard DTD union mechanism can be used to do this.

2. The following steps describe a second approach that involves creating a separate DTD and defines the structure used at runtime for the ANY tag.

   a. Import the DTD for the event, while creating an Application Data Type or creating the published/subscribed event or the invoked/implemented procedure. iStudio warns about the ANY tag and points out the type that needs to be modified.

   b. Reload the iStudio project.

   c. Under the list of ADTs, find the type corresponding to the ANY element and right-click to display the context menu. This is the ADT mentioned in Step a.

   d. Import a DTD, which defines the structure planned to use, for the ANY tag.

   This method does not support using PCDATA tag for the ANY element. The ANY element must have a sub-element in this case.

   For example, consider the following, customer.dtd:

```
<!ELEMENT customer (name, phone, address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT address ANY)>
```

   When this DTD is imported, iStudio warns that the address tag is an ANY tag and it corresponds to the address ADT in iStudio.

   The address_any.dtd could look like the following:

```
<!ELEMENT address_any (street, city, zip)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT zip ANY)>
```

Import the `address_any.dtd` by right-clicking the address ADT in iStudio. This assumes that the XML has an `address_any` element under the address element, as follows:

```
<address>
  <address_any>
    <street>
    <city>
    <zip>
  </address_any>
</address>
```

If the `address_any` element is not needed, then instead of editing the address ADT, edit customer ADT and change the type of address attribute from `address` to `address_any`, after importing `address_any` elsewhere. The following is now true:

```
<address>
  <street>
  <city>
  <zip>
</address>
```

A.11 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

Adapter	Entry In Oracle Wallet
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
  ```

Index