Oracle® Application Server Integration InterConnect

Adapter for AQ Installation and User's Guide

10*g* Release 2 (10.1.2)

Part No. B14077-01

November 2004

ORACLE®

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

Part No. B14077-01

Primary Author: Vimmy K Raj, Pradeep Vasudev

Contributing Author: Ashwin Patel, Maneesh Joshi, Rahul Pathak, Harish Sriramulu

Contents

Send Us Your Comments

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

Part No. B14077-01

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: appserverdocs_us@oracle.com
- FAX: 650-506-7375 Attn: Oracle Application Server Documentation Manager
- Postal service:

  Oracle Corporation
  Oracle Application Server Documentation Manager
  500 Oracle Parkway, M/S 1op6
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This Preface contains these topics:

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. 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 additional information, visit the Oracle Accessibility Program Web site at

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

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, 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, JAWS 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.

Structure

This document contains:

Chapter 1, "Introduction"

This chapter describes the Oracle Application Server Integration InterConnect Adapter for Advanced Queuing (AQ adapter), and the hardware and software requirements.

Chapter 2, "Installation and Configuration"

This chapter describes installation and configuration of the AQ adapter.

Chapter 3, "Design Time and Runtime Concepts"

This chapter describes the design time and runtime concepts of the AQ adapter.

Chapter 4, "Sample Use Cases"

This chapter provides sample use cases for the AQ adapter.

Appendix A, "Frequently Asked Questions"

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

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*

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://www.oracle.com/technology/membership/`

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://www.oracle.com/technology/documentation/`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text

Conventions in Text

We use the following conventions in text to help you more quickly identify special terms. The table also provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle Database 10g Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width) font`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, Recovery Manager keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executable programs, filenames, directory names, and sample user-supplied elements. *Note:* Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to start SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run *`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
`[ ]`	Anything enclosed in brackets is optional.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
`{ }`	Braces are used for grouping items.	`{ENABLE	DISABLE}`

Convention	Meaning	Example
|	A vertical bar represents a choice of two options.	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	Ellipsis points mean repetition in syntax descriptions.	CREATE TABLE ... AS *subquery*;
	In addition, ellipsis points can mean an omission in code examples or text.	SELECT *col1*, *col2*, ... , *coln* FROM employees;
Other symbols	You must use symbols other than brackets ([ ]), braces ({ }), vertical bars (|), and ellipsis points (...) exactly as shown.	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*
UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. Because these terms are not case sensitive, you can use them in either UPPERCASE or lowercase.	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
lowercase	Lowercase typeface indicates user-defined programmatic elements, such as names of tables, columns, or files. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;

Conventions for Windows Operating Systems

The following table describes conventions for Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Click **Start**, and then choose the ***menu item***	How to start a program.	To start the Database Configuration Assistant, click **Start**, and choose **Programs**. In the Programs menu, choose **Oracle** - ***HOME_NAME*** and then click **Configuration and Migration Tools**. Choose **Database Configuration Assistant**.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the filename begins with \\, then Windows assumes it uses the Universal Naming Convention.	c:\winnt"\"system32 is the same as C:\WINNT\SYSTEM32

Convention	Meaning	Example
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`
Special characters	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	`C:\>exp HR/HR TABLES=employees QUERY=\"WHERE job_id='SA_REP' and salary<8000\"`
HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*HOME_NAME*`TNSListener`
ORACLE_HOME and *ORACLE_BASE*	In releases prior to Oracle8*i* release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory. This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle\product\10.1.0`. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\product\10.1.0\db_`*n*, where *n* is the latest Oracle home number. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle Database Installation Guide for Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\admin` directory.

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:

- Advanced Queuing Adapter Overview
- Advanced Queuing Adapter System Requirements

1.1 Advanced Queuing 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.

The purpose of this guide is to explain all the necessary design time and runtime concepts of the AQ adapter.

1.2 Advanced Queuing 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 the computer where the AQ adapter will be installed.

Table 1–1 Hardware Requirements

Hardware	Windows 2000	UNIX
Disk Space	500 MB	500 MB
Memory	128 MB	128 MB

1.2.2 Software Requirements

The following sections describe the AQ adapter software requirements:

- Operating System Requirements

Operating System Requirements

Table 1–2 lists the operating system requirements for the computer where the AQ adapter will be installed.

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:

2.1 Installing the Advanced Queuing 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:

2.1.1 Preinstallation Tasks

Consult the following guides before installing the AQ adapter:

- *Oracle Application Server Installation Guide* for information about OUI startup.
- *Oracle Application Server InterConnect Installation Guide* for information on mounting CD-ROMs, software, hardware, and system requirements for OracleAS Integration InterConnect.

2.1.2 Installation Tasks

To install the AQ adapter:

1. Select **AQ adapter** in the Available Product Components page of the OracleAS Integration InterConnect installation, and click **Next**.
2. The Set Oracle Wallet Password screen is displayed. Enter and confirm the password on the screen, which will be used to administer OracleAS Integration InterConnect installation. Click **Next**.
    - Go to step 3, 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 4, 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.

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 System Identifier (SID) for the hub database.
   - Password: The user password for the hub database user.
4. Click **Next**. The Specify AQ Adapter Name page is displayed.
5. Enter the application name. Blank spaces are not permitted. The default value is `myAQApp`.
6. Click **Next**. The Application Spoke Database: Specify Database Connection Information page is displayed. This page 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 page is for Advanced Queues, from which the adapter will deliver or receive messages.
7. Click **Next**. The Spoke Application Database AQ Username page 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.
8. Click **Next**. The Summary page is displayed.
9. 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 5.

10. Click **Exit** on the Installation page to exit the AQ adapter installation.

2.2 Configuring the Advanced Queuing Adapter

After an AQ adapter installation, you can configure it for your needs. 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 Advanced Queuing 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 `oailog.txt` log file.
`persistence`	The messages are persisted in this directory. Do not edit this directory or its files.

2.2.1 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 runtime 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 4 on page 2-2

2.2.2 Ini File Settings

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

- hub.ini File
- adapter.ini File

2.2.2.1 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 parameters, their 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. The number is between 2 and the number of nodes. For example, if the cluster contains 4 nodes, `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`

2.2.2.2 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`

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 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
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 that 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 (like '%,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 turn on 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 turn on 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,%'`

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=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 turn on 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`
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 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:\InterConnect10_1_2\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 for the date format only.	`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 for the date format only.	`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`

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

Parameter	Description	Example
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 for Microsoft Windows only.	`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 for Microsoft Windows only.	`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 for Microsoft Windows only.	`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 for Microsoft Windows only.	`service_max_heap_size=536870912`
service_max_java_stack_size	Specifies the maximum size the JVM stack can grow. Possible Value: A valid JVM maximum stack size. Default Value: Default value for the JVM. This parameter is for Microsoft Windows only.	`service_max_java_stack_size=409600`
service_max_native_stack_size	Specifies the maximum size 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 for Microsoft Windows only.	`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 for Microsoft Windows only.	`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 for Microsoft Windows only.	`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 for Microsoft Windows only.	`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 for Microsoft Windows only.	`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`

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

Advanced Queuing Adapter-specific Parameters

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

Table 2–9 Advanced Queuing 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
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 How do I secure my passwords?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

Table 2–9 (Cont.) Advanced Queuing Adapter-specific Parameters

Parameter	Description	Example
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.agen t.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 Runtime Concepts

This chapter describes the design time and runtime concepts for the AQ adapter. It contains the following topics:

3.1 Advanced Queuing Adapter Design Time Concepts

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

3.1.1 RAW Payload with XML data

You can use iStudio to import a Document Type Definition (DTD) 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 wish to send two objects, `Customer` and `PurchaseOrder`, as part of one OracleAS Integration InterConnect message. 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 datatypes 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 datatype of the `Customer` attribute from string to the attribute created when `customer.dtd` was imported. Similarly, change the datatype 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 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 container on the deploy tab in iStudio.
  2. Expand applicationName.
  3. Expand the Routing container.
  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 Advanced Queuing Adapter Runtime Concepts

This section describes the runtime 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 runtime 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.

3.4 Starting the Advanced Queuing 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. The Services window is displayed.

On...	Choose...
Windows 2000	Start, Settings, Control Panel, Administrative Tools, Services

  2. Select the **OracleHomeOracleASInterConnectAdapter-Application** service.
  3. Start the service based on the operating system.

On...	Choose...
Windows 2000	Right-click the service and select Start from the context menu.

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 Advanced Queuing Adapter Instance

You can verify the start up status of the AQ adapter by viewing the `oailog.txt` files. The files are located in the timestamped 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 Advanced Queuing 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. The Services window is displayed.

On...	Choose...
Windows 2000	Start, Settings, Control Panel, Administrative Tools, Services

  2. Select the **OracleHomeOracleASInterConnectAdapter-Application** service.
  3. Stop the service based on the operating system.

On...	Choose...
Windows 2000	Right-click the service and select Stop from the context menu.

You can verify the stop status of the AQ adapter by viewing the `oailog.txt` files. These files are located in the timestamped 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. The following 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. On the Create Business Object dialog, enter customer in the Business Object Name field.
2. Create the event in iStudio. On the Create Event dialog, complete the following:
   a. Select customer for the 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. On the Create Application dialog, enter aqapp in the Application Name field.
4. Create a Published Event using the Publish Event Wizard in iStudio:
   a. On the Select an Event page:
      * 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 page:
      * 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 page.

d. Click **Finish**.

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

a. On the Select an Event page:

* 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 Page:

* Click **Import** and select **XML** on the Define Application View page 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 page.

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 Runtime 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. On the Create Business Object dialog, enter customer in the Business Object Name field.
5. Create a Procedure in iStudio. On the Create Procedure dialog, complete the following:
   a. Select customer for 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. On the Create Data Type dialog, 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 page:
      * 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 page:
      * Click **Import** and select **Database**.
      * Log in as the system user on the Database Login dialog.
      * On the Oracle Database Browser dialog, expand the AQUSER list and select AQUSER.MY_QUEUE_TYPE.
      * Click **Done**.
   c. Create the following mapping on the Define Mapping IN Arguments page:
      * For aqapp view to the common view: `updateCustomer:IN - ObjectCopy - updateCustomer:IN`
   d. Create the following mapping on the Define Mapping OUT Arguments page:

* 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 Runtime 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 Advanced Queuing adapter did not start properly. What is wrong?
- Why is the AQ adapter using old information after I changed information in iStudio?
- Which databases are referred to during installation?
- What is the 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?
- Can I install multiple Advanced Queuing adapters on the same computer?
- How do I handle `ANY` tags in DTDs imported into iStudio?
- How do I secure my passwords?

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

View the `oailog.txt` file located in the timestamped 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 4 on page 2-2, and *timestamp_in_milliseconds* is the directory. If no exceptions are listed, then the adapter has started properly.

The Advanced Queuing adapter did not start properly. What is wrong?

View the exceptions in the AQ adapter log file (`oailog.txt`). 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

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.

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.

What is the 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.

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 IC 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-separated consumer names.

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

Platform	Directory
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

Can I install multiple Advanced Queuing adapters on the same computer?

The installer overwrites previous installations of the AQ adapter if you try to install it a second time in the same Oracle home. However, you can have multiple Oracle homes on a computer and have one AQ adapter in each Oracle home. When you install the AQ adapter a second time, choose a different Oracle home.

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>
```

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
```

For example,

```
AQAPP/aq_bridge_schema_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.

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

A

B

C

D

E

H

I

J

L

M

N

O

P

R

S