

Oracle® Application Server InterConnect

Adapter for AQ Installation and User's Guide

10g (9.0.4)

Part No. B10416-01

September 2003

Oracle Application Server InterConnect Adapter for AQ Installation and User's Guide, 10g (9.0.4)

Part No. B10416-01

Copyright © 2002, 2003 Oracle Corporation. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent and other intellectual and industrial property laws. Reverse engineering, disassembly or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and Oracle*MetaLink*, Oracle Store, Oracle9i, Oracle Discoverer, SQL*Plus, and PL/SQL are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Send Us Your Comments	vii
Preface.....	ix
Audience	x
Documentation Accessibility	x
Organization.....	x
Related Documentation	xi
Conventions.....	xii
1 Introduction	
Advanced Queuing Adapter Overview.....	1-2
Hardware Requirements	1-2
Software Requirements.....	1-2
Operating System Requirements	1-2
JRE Requirements.....	1-3
Database Requirements.....	1-3
2 Installation and Configuration	
Installing the Advanced Queuing Adapter	2-2
Preinstallation Tasks	2-2
Installation Tasks.....	2-3
Configuring the Advanced Queuing Adapter	2-5
Using the Application Parameter.....	2-6
Ini File Settings.....	2-7

Hub.ini Parameters.....	2-7
RAC-specific Hub.ini Parameters	2-8
Adapter.ini Parameters.....	2-9
Advanced Queuing Adapter Parameters	2-17
Starting the Advanced Queuing Adapter	2-19
Stopping the Advanced Queuing Adapter	2-20

3 Design Time and Runtime Concepts

Advanced Queuing Adapter Design Time Concepts.....	3-2
Supported Features	3-2
RAW Payload.....	3-2
Oracle Object Payload with and without XML Data.....	3-2
Creating Metadata in iStudio	3-3
Importing from XML:	3-4
Returned In Arguments.....	3-4
Advanced Queuing Adapter Runtime Concepts	3-5
How the Advanced Queuing Adapter Works.....	3-5
Sending Adapter.....	3-5
Receiving Adapter	3-6
Starting the Advanced Queuing Adapter	3-6
Sample Log File of Successfully Started Advanced Queuing Adapter	3-7
Stopping the Advanced Queuing Adapter	3-8

4 Sample Use Cases

Advanced Queuing Adapter Sample Use Cases.....	4-2
Case One: The Advanced Queuing Publishes and Subscribes with RAW Payload	4-2
Design Time Steps	4-2
Runtime Steps:	4-3
Related Files	4-4
Case Two: The Advanced Queuing Invokes and Implements with Oracle Object Type Payload 4-5	
Design Time Steps	4-6
Runtime Steps	4-7
Related Files	4-8

5 Frequently Asked Questions

Installation Questions	5-2
Design Time Questions	5-4

Index

Send Us Your Comments

Oracle Application Server InterConnect Adapter for AQ Installation and User's Guide, 10g (9.0.4)

Part No. B10416-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. 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?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, 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 10p6
Redwood Shores, CA 94065 USA

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

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

Preface

This preface contains these topics:

- [Audience](#)
- [Documentation Accessibility](#)
- [Organization](#)
- [Related Documentation](#)
- [Conventions](#)

Audience

This book is intended for those who perform the following tasks:

- install applications
- maintain applications

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 Corporation 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 Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Organization

This document contains:

Chapter 1, "Introduction"

This chapter describes the Oracle Application Server InterConnect Advanced Queuing adapter and the hardware and software requirements.

Chapter 2, "Installation and Configuration"

This chapter describes installation and configuration of the Advanced Queuing adapter.

Chapter 3, "Design Time and Runtime Concepts"

This chapter describes the design time and runtime concepts for the Advanced Queuing adapter.

Chapter 4, "Sample Use Cases"

This chapter provides sample use cases for the Advanced Queuing adapter.

Chapter 5, "Frequently Asked Questions"

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

Related Documentation

For more information, see these Oracle resources:

- *Oracle Application Server InterConnect User's Guide*
- *Oracle Application Server InterConnect Installation Guide*
- *Oracle Application Server InterConnect Adapter Configuration Editor User's 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://otn.oracle.com/membership/>

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

<http://otn.oracle.com/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 Code Examples](#)
- [Conventions for Microsoft Windows Operating Systems](#)

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and 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 .
<i>Italics</i>	Italic typeface indicates book titles or emphasis.	<i>Oracle9i Database Concepts</i> Ensure that the recovery catalog and target database do <i>not</i> 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, RMAN 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.

Convention	Meaning	Example
lowercase monospace (fixed-width font)	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter <code>sqlplus</code> to open SQL*Plus. The password is specified in the <code>orapwd</code> file. Back up the datafiles and control files in the <code>/disk1/oracle/dbs</code> directory. The <code>department_id</code> , <code>department_name</code> , and <code>location_id</code> columns are in the <code>hr.departments</code> table. Set the <code>QUERY_REWRITE_ENABLED</code> initialization parameter to <code>true</code> . Connect as <code>oe</code> user. The <code>JRepUtil</code> class implements these methods.
<i>lowercase monospace (fixed-width font) italic</i>	Lowercase monospace italic font represents placeholders or variables.	You can specify the <i>parallel_clause</i> . Run <code>Uold_release.SQL</code> where <i>old_release</i> 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
[]	Brackets enclose one or more optional items. Do not enter the brackets.	<code>DECIMAL (digits [, precision])</code>
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	<code>{ENABLE DISABLE}</code>
	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	<code>{ENABLE DISABLE}</code> <code>[COMPRESS NOCOMPRESS]</code>

Convention	Meaning	Example
...	Horizontal ellipsis points indicate either: <ul style="list-style-type: none"> That we have omitted parts of the code that are not directly related to the example That you can repeat a portion of the code 	<pre>CREATE TABLE ... AS subquery; SELECT col1, col2, ... , coln FROM employees;</pre>
.	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	<pre>acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;</pre>
<i>Italics</i>	Italicized text indicates placeholders or variables for which you must supply particular values.	<pre>CONNECT SYSTEM/system_password DB_NAME = database_name</pre>
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. However, because these terms are not case sensitive, you can enter them in lowercase.	<pre>SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;</pre>
lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files. Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	<pre>SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;</pre>

Conventions for Microsoft Windows Operating Systems

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

Convention	Meaning	Example
Choose Start >	How to start a program.	To start the Oracle Database Configuration Assistant, choose Start > Programs > Oracle - <i>HOME_NAME</i> > Configuration and Migration Tools > 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 file name begins with \\, then Windows assumes it uses the Universal Naming Convention.	c:\winnt "\"system32 is the same as C:\WINNT\SYSTEM32
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 <i>command prompt</i> in this manual. 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:\oracle\oradata> C:\>exp scott/tiger TABLES=emp QUERY=\"WHERE job='SALESMAN' and sal<1600\" C:\>imp SYSTEM/password FROMUSER=scott TABLES=(emp, dept)
<i>HOME_NAME</i>	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 <i>HOME_NAME</i> TNSListener

Convention	Meaning	Example
<i>ORACLE_HOME</i> and <i>ORACLE_BASE</i>	<p>In releases prior to Oracle8i release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level <i>ORACLE_HOME</i> directory that by default used one of the following names:</p> <ul style="list-style-type: none"> ■ C:\orant for Windows NT ■ C:\orawin95 for Windows 95 ■ C:\orawin98 for Windows 98 <p>This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level <i>ORACLE_HOME</i> directory. There is a top level directory called <i>ORACLE_BASE</i> that by default is C:\oracle. If you install Oracle9i release 1 (9.0.1) on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is C:\oracle\ora90. The Oracle home directory is located directly under <i>ORACLE_BASE</i>.</p> <p>All directory path examples in this guide follow OFA conventions.</p> <p>Refer to <i>Oracle9i Database Getting Starting for Windows</i> for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.</p>	Go to the <i>ORACLE_BASE\ORACLE_HOME\rdms\admin</i> directory.

1

Introduction

This chapter provides an introduction to the Oracle Application Server InterConnect Adapter for AQ (Advanced Queuing adapter) and how the adapter works with Oracle Application Server InterConnect.

Advanced Queuing Adapter Overview

The Oracle Application Server InterConnect Advanced Queuing adapter enables an Oracle Advanced Queuing application to be integrated with other applications using Oracle Application Server InterConnect (OracleAS InterConnect). This adapter is useful in all enterprise application integration scenarios involving Oracle Database Advanced Queuing applications. The purpose of this guide is to explain all the necessary design time and runtime concepts of the Advanced Queuing adapter.

Hardware Requirements

[Table 1-1](#) lists the hardware requirements for the computer on which the Oracle Advanced Queuing adapter is installed.

Table 1-1 Hardware Requirements

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

Software Requirements

The following are software requirements for the Advanced Queuing adapter:

- [Operating System Requirements](#)
- [JRE Requirements](#)

Operating System Requirements

[Table 1-2](#) lists operating system requirements for the computer on which the Advanced Queuing adapter is installed.

Table 1-2 Operating System Requirements

Operating System	Version
Windows 2000	service pack 1 or above
Windows NT 4.0	service pack 6 or above
IBM AIX 5L	5.1 and 5.2 (64 bit)
HP Tru64	5.1a and 5.1b (64 bit)

Table 1–2 Operating System Requirements

Operating System	Version
HP-UX	11.0 and 11.11 (64 bit)
Red Hat Linux	Advanced Server 2.1
Sun SPARC Solaris	8 and 9 (32 bit)

JRE Requirements

Oracle Application Server InterConnect uses JRE 1.4.1 which is installed with its components.

Database Requirements

The Oracle Database 8i and later versions support the Advanced Queuing adapter feature. The Oracle Advanced Queuing Application does not need to be installed on the spoke machine. The Advanced Queuing adapter can be installed on a remote machine.

Installation and Configuration

This chapter describes installation and configuration of the Advanced Queuing adapter. This chapter discusses the following topics:

- [Installing the Advanced Queuing Adapter](#)
- [Configuring the Advanced Queuing Adapter](#)
- [Starting the Advanced Queuing Adapter](#)
- [Stopping the Advanced Queuing Adapter](#)

Installing the Advanced Queuing Adapter

This section describes the following topics:

- [Preinstallation Tasks](#)
- [Installation Tasks](#)

Preinstallation Tasks

The Advanced Queuing adapter must be installed in one of the following Oracle homes:

- An existing OracleAS InterConnect Oracle home for 10g (9.0.4)
- A new Oracle home (the installer creates this for you)

Consult the following guides before proceeding with Advanced Queuing adapter installation:

- *Oracle Application Server InterConnect Installation Guide*, which includes information on:
 - OracleAS InterConnect software, hardware, and system requirements
 - OracleAS InterConnect installation
- *Oracle Application Server Installation Guide*, which includes information on:
 - Oracle Universal Installer startup

Note: OracleAS InterConnect Hub is installable through the OracleAS InterConnect Hub installation type. You must install the OracleAS InterConnect Hub before proceeding with the Advanced Queuing adapter installation.

Installation Tasks

To install the Advanced Queuing adapter, start the installer and complete the following steps:

1. On the Available Product Components page of the OracleAS InterConnect installation, select Advanced Queuing adapter, then select **Next**.

Consider the following scenarios:

- If installing the Advanced Queuing adapter in an independent Oracle home, make sure that the OracleAS InterConnect Hub has been installed, not necessarily in the same Oracle home. Continue to step 2.
- If installing the Advanced Queuing adapter in an existing Oracle home, make sure that it is a home directory to one of the OracleAS InterConnect component. Continue to step 3.

Note: The hub database information, such as the SID, host, port, and username/password from the Hub installation, is needed for step 2.

2. If installing OracleAS InterConnect for the first time on this machine, complete the following steps to enter the hub database information:
 - a. On the Available Products page, select **Next**. The Database Configuration page displays. Enter information in the following fields:
 - * Host Name—The host name of the machine where the hub database is installed.
 - * Port Number—The TNS listener port for the hub database.
 - * Database SID—The SID for the hub database.
 - b. Click **Next**. The Database User Configuration page displays. Enter information in the following fields:
 - * User Name—The hub database user name. Make sure the OracleAS InterConnect Hub is installed. If the Hub is not installed, complete the installation and note the user name and password.
 - * Password—The password for the hub database user.

3. Click **Next**. The Adapter Configuration page displays. Enter the application to be defined or already defined in iStudio in the Application field. White spaces or blank spaces are not permitted. The default value is `myAQAPP`.
4. Select **Next**. The Spoke Application Database Configuration page displays. Enter information in the following fields:
 - Host Name—The name of the machine where the application database is installed.
 - Port Number—The database TNS listener port.
 - Database SID—The SID for the application database.

The information on this page is for the database on the application side from which the adapter will put or receive messages. This is not the information for the hub database.

5. Select **Next**. The Spoke Application Database User Configuration page displays. Enter information in the following fields:
 - User Name—The name the Advanced Queuing adapter uses to connect to the database.
 - Password—The password for the user name.
 - Consumer Name—The application that writes to the queue uses a consumer name to indicate that OracleAS InterConnect should pick up a message. Alternatively, the Advanced Queuing adapter may have a subscriber configured to which the Advanced Queuing adapter corresponds.

Leave this field blank if the queues the Advanced Queuing adapter will connect to on the application database side are single consumer queues. However, if any of the queues are multi-consumer queues, 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 written, look at the code or the documentation that comes with it to determine the consumer name. For example, for iProcurement to SAP integration, the consumer name can be found in the iProcurement to SAP documentation.

- * If the code that will write a message to the queue is not written, 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, use the database's Advanced Queuing APIs to find the name.
6. Select **Next**. Complete the fields for any other components selected for installation, such as other adapters. When finished, the Summary page displays.
 7. Select **Install** to install the Advanced Queuing adapter and other selected components. The Advanced Queuing adapter is installed in the following directory:

Platform	Directory
Windows	<code>ORACLE_HOME\oai\9.0.4\adapters\Application</code>
UNIX	<code>ORACLE_HOME/oai/9.0.4/adapters/Application</code>

Configuring the Advanced Queuing Adapter

[Table 2–2](#), [Table 2–3](#), and [Table 2–4](#) describe executable files, configuration files, and directories. These files and directories are accessible from the directory shown in [Table 2–1](#):

Table 2–1 *Advanced Queuing Adapter Directory*

Platform	Directory
UNIX	<code>ORACLE_HOME/oai/9.0.4/adapters/Application</code>
Windows	<code>ORACLE_HOME\oai\9.0.4\adapters\Application</code>

Table 2–2 *Executable Files*

File	Description
<code>start.bat</code> (Windows)	Takes no parameters, starts the adapter.
<code>start</code> (UNIX)	Takes no parameters, starts the adapter.
<code>stop.bat</code> (Windows)	Takes no parameters; stops the adapter.
<code>stop</code> (UNIX)	Takes no parameters; stops the adapter.

Table 2–2 Executable Files

File	Description
<code>ignoreErrors.bat</code> (Windows)	If an argument is specified, then the given error code will be ignored. If no argument is specified, than all error codes specified in the <code>ErrorCodes.ini</code> will be ignored.
<code>ignoreErrors</code> (UNIX)	If an argument is specified, then the given error code will be ignored. If no argument is specified, than all error codes specified in the <code>ErrorCodes.ini</code> will be ignored.

Table 2–3 Configuration Files

File	Description
<code>ErrorCodes.ini</code> (Windows and UNIX)	Contains one error code per line.
<code>adapter.ini</code> (Windows and UNIX)	Consists of all the initialization parameters which the adapter reads at startup. Refer to Appendix A for a typical <code>adapter.ini</code> file.

Table 2–4 Directories

File	Description
<code>persistence</code>	The messages are persisted in this directory. Do not edit this directory or its files.
<code>logs</code>	The logging of adapter activity is done in subdirectories of the log directory. Each new run of the adapter creates a new subdirectory in which logging is done in an <code>oailog.txt</code> file.

Using the Application Parameter

Adapters do not have integration logic. The Advanced Queuing adapter has a generic transformation engine that processes metadata from the repository as runtime instructions to do transformations. The application defines for an adapter what its capabilities are. For example, it can define what messages it can publish, what messages it can subscribe to, and what are the transformations to perform. The application parameter allows the adapter to become smart in the context of the application to which it is connected. It allows the adapter to retrieve from the repository only that metadata that is relevant to the application. The application parameter must match the corresponding application that will be defined in *iStudio* under the Applications folder.

If you are using pre-packaged metadata, after importing the pre-packaged metadata into the repository, start up *iStudio* to find the corresponding application (under the Applications folder in *iStudio*) to use as the application for the adapter you are installing (unless the package you are using provides directions for what the application should be).

Ini File Settings

This section contains these topics:

- [Hub.ini Parameters](#)
- [RAC-specific Hub.ini Parameters](#)
- [Adapter.ini Parameters](#)
- [Advanced Queuing Adapter Parameters](#)

Hub.ini Parameters

The Advanced Queuing adapter connects to the hub database using parameters from the `hub.ini` file located in the hub directory. [Table 2-5](#) lists the parameter name, a description for each parameter, the possible and default values, and an example.

Table 2-5 *Hub.ini Parameters*

Parameters	Description	Example
<code>hub_username</code>	The name of the hub database schema (or username). Possible values are valid hub database username. There is no default value.	<code>hub_username=myhub</code>
<code>hub_password</code>	The password for the hub database user. Possible values are the valid password for the hub database user. There is no default value.	<code>hub_password=manager</code>
<code>hub_host</code>	The name of the machine hosting the hub database. Possible values are the valid machine name. There is no default value.	<code>hub_host=mpjoshipc</code>

Table 2–5 Hub.ini Parameters

Parameters	Description	Example
hub_instance	The valid SID of the hub database. There is no default value.	hub_instance=orcl
hub_port	The TNS listener port number for the HUB database instance. There is no default value.	hub_port=1521
repository_name	The valid name of the repository this adapter talks to. The default value is InterConnectRepository.	repository_name=InterConnectRepository

RAC-specific Hub.ini Parameters

When a hub is installed on a Real Application Cluster (RAC) database, parameters listed in [Table 2–6](#) represent information on additional nodes used for connection and configuration. These parameters are added on top of the default parameters which represent the primary node. In [Table 2–6](#), *x* represent the node number, which varies between 2 and the number of nodes. For example, if the RAC setup contains 4 nodes, *x* can take a value between 2 and 4.

Table 2–6 RAC-specific Hub.ini Parameters

Parameter	Description	Example
hub_num_nodes	Number of nodes in RAC cluster.	hub_num_nodes=4
hub_hostx	This parameter represents the host where the RAC database is installed.	hub_host2=dsunram13
hub_instancex	This parameter represents the instance on the respective node.	hub_instance2=orcl2
hub_portx	This parameter represents the port where the node has its instance available.	hub_port2=1521

Adapter.ini Parameters

The Advanced Queuing adapter connects to the spoke application using parameters from the `adapter.ini` file. [Table 2-7](#) lists the parameter name, a description for each parameter, the possible and default values and an example.

Table 2-7 Adapter.ini Parameters

Parameter	Description	Example
<code>application</code>	The name of the application this adapter connects to. This must match with the name specified in iStudio during creating of metadata. Any alphanumeric string can be used. There is no default value.	<code>application=aqapp</code>
<code>partition</code>	The partition this adapter handles as specified in iStudio. Any alphanumeric string is a possible value. There is no default value.	<code>partition=germany</code>
<code>instance_number</code>	To have multiple adapter instances for the given application with the given partition, each adapter should have a unique instance number. Possible values are any integer greater than or equal to 1. There is no default value.	<code>instance_number=1</code>
<code>agent_log_level</code>	Specifies the amount of logging necessary. Possible values are: 0=errors only 1=status and errors 2=trace, status, and errors The default value is 1.	<code>agent_log_level=2</code>
<code>agent_subscriber_name</code>	The subscriber name used when this adapter registers its subscription. The possible value is a valid Oracle Advanced Queuing subscriber name and there is no default value.	<code>agent_subscriber_name=aqapp</code>
<code>agent_message_selector</code>	Specifies conditions for message selection when registering its subscription with the hub. The possible value is a valid Oracle Advanced Queuing message selector string. There is no default value.	<code>agent_message_selector=recipient_list like '%aqapp,%'</code>
<code>agent_reply_subscriber_name</code>	The subscriber name used when multiple adapter instances for the given application with the given partition are used. Optional if there is only one instance running. The possible value is application name (parameter: <code>application</code>) concatenated with instance number (parameter: <code>instance_number</code>). There is no default value.	If <code>application=aqapp</code> , <code>instance_number=2</code> , then, <code>agent_reply_subscriber_name=aqapp2</code>

Table 2–7 Adapter.ini Parameters

Parameter	Description	Example
agent_reply_ message_selector	Used only if multiple adapter instances for the given application with the given partition. The possible value is a string built using concatenating application name (parameter:application) with instance number (parameter:instance_number). There is no default value.	If application=aqapp, instance_number=2, then agent_reply_message_selector=recipient_list like '%,aqapp2,%'
agent_tracking_ enabled	Specifies if message tracking is enabled. Set to false to turn off all tracking of messages. Set to true to track messages with tracking fields set in iStudio. Possible values are true or false. The default value is true.	agent_tracking_enabled=true
agent_ throughput_ measurement_ enabled	Specifies if throughput measurement is enabled. Set to true to turn on all throughput measurements. Possible values are true or false. The default value is true.	agent_throughput_measurement_enabled=true
agent_use_ custom_hub_dtd	Specifies if a custom DTD should be used for the common view message when handing it to the hub. By default adapters use an OracleAS InterConnect-specific DTD for all messages sent to the hub as other OracleAS InterConnect adapters will be retrieving the messages from the hub and know how to interpret them. Set to true if for every message, the DTD imported for the message of the common view is to be used instead of the OracleAS InterConnect DTD. Only set to true if a OracleAS InterConnect adapter is not receiving the messages from the hub. Possible values are true or false. There is no default value.	agent_use_custom_hub_dtd=false
agent_metadata_ caching	Specifies the metadata caching algorithm. Possible values are: <ul style="list-style-type: none"> ▪ startup—Cache everything at startup. This may take a while if there are a lot of tables in the repository. ▪ demand—Cache metadata as it is used. ▪ none—No caching. This slows down performance. The default value is demand.	agent_metadata_caching=demand

Table 2–7 Adapter.ini Parameters

Parameter	Description	Example
agent_dvm_table_caching	<p>Specifies the domain value mapping (DVM) table caching algorithm. Possible values are:</p> <ul style="list-style-type: none"> ▪ startup—Cache all DVM tables at startup. This may take a while if there are a lot of tables in the repository. ▪ demand—Cache tables as they are used. ▪ none—No caching. This slows down performance. <p>The default value is demand.</p>	agent_dvm_table_caching=demand
agent_lookup_table_caching	<p>Specifies the lookup table caching algorithm. Possible values are:</p> <ul style="list-style-type: none"> ▪ startup—Cache all lookup tables at startup. This may take a while if there are a lot of tables in the repository. ▪ demand—Cache tables as they are used. ▪ none—No caching. This slows down performance. <p>The default value is demand.</p>	agent_lookup_table_caching=demand
agent_delete_file_cache_at_startup	<p>With any of the agent caching methods enabled, metadata from the repository is cached locally on the file system.</p> <p>Set this parameter to true to delete all cached metadata on startup.</p> <p>Note: After changing metadata or DVM tables for this adapter in iStudio, you must delete the cache to guarantee access to the new metadata or table information.</p> <p>Possible values are true or false. The default value is false.</p>	agent_delete_file_cache_at_startup=false
agent_max_ao_cache_size	<p>Specifies the maximum number of application objects' metadata to cache. Possible values are any integer greater than or equal to 1. The default value is 200.</p>	agent_max_ao_cache_size=200
agent_max_co_cache_size	<p>Specifies the maximum number of common objects' metadata to cache. Possible values are any integer greater than or equal to 1. The default value is 100.</p>	agent_max_co_cache_size=100
agent_max_message_metadata_cache_size	<p>Specifies the maximum number of messages' metadata to cache (publish/subscribe and invoke/implement). Possible values are any integer greater than or equal to 1. The default value is 200.</p>	agent_max_message_metadata_cache_size=200

Table 2–7 Adapter.ini Parameters

Parameter	Description	Example
agent_max_dvm_table_cache_size	Specifies the maximum number of DVM tables to cache. Possible values are any integer greater than or equal to 1. The default value is 200.	agent_max_dvm_table_cache_size=200
agent_max_lookup_table_cache_size	Specifies the maximum number of lookup tables to cache. Possible values are any integer greater than or equal to 1. The default value is 200.	agent_max_lookup_table_cache_size=200
agent_max_queue_size	Specifies the maximum size that internal OracleAS InterConnect message queues can grow. Possible values are any integer greater than or equal to 1. The default value is 1000.	agent_max_queue_size=1000
agent_persistence_queue_size	Specifies the maximum size that internal OracleAS InterConnect persistence queues can grow. Possible values are any integer greater than or equal to 1. The default value is 1000.	agent_persistence_queue_size=1000
agent_persistence_cleanup_interval	Specifies how often the persistence cleaner thread should run. Possible values are any integer greater than or equal to 30000 milliseconds. The default value is 60000.	agent_persistence_cleanup_interval=60000
agent_persistence_retry_interval	Specifies how often the persistence thread should retry when it fails to push a OracleAS InterConnect message. Possible values are any integer greater than or equal to 5000 milliseconds. The default value is 60000.	agent_persistence_retry_interval=60000
agent_pipeline_to_hub	Specifies how to turn on or off the pipeline for messages from the Bridge towards the hub. If you set the pipeline to <i>false</i> , the file persistence is not used in that direction.	agent_pipeline_to_hub=false
agent_pipeline_from_hub	Specifies how to turn on or off the pipeline for messages from the hub towards the Bridge. If you set the pipeline to <i>false</i> , the file persistence is not used in that direction.	agent_pipeline_from_hub=false
service_path	Windows only. The value that the environment variable PATH should be set to. path is set to the specified value before forking the Java VM. Typically, all directories containing all necessary DLLs should be listed here. Possible values are the valid path environment variable setting. There is no default value.	service_path=%JREHOME%\bin;D:\oracle\ora904\bin

Table 2-7 Adapter.ini Parameters

Parameter	Description	Example
service_classpath	The classpath used by the adapter Java VM. If a custom adapter is developed and as a result, the adapter is to be used to pick up any additional jars, add the jars to the existing set of jars being picked up. Possible values are the valid classpath. There is no default value.	service_classpath=D:\oracle\ora904\oai\904\lib\oai.jar;%JREHOME%\lib\i18n.jar; D:\oracle\ora904\jdbc\classes12.zip
service_class	Specifies the entry class for the Windows service. A possible value is oracle/oai/agent/service/AgentService. There is no default value.	service_class=oracle/oai/agent/service/AgentService
service_max_java_stack_size	Windows only. The maximum size to which the Java VM's stack can grow. Possible values are the valid Java VM maximum native stack size. The default value is the default for the Java VM.	service_max_java_stack_size=409600
service_max_native_stack_size	Windows only. The maximum size to which the Java VM's native stack can grow. Possible values are the valid Java VM maximum native stack size. The default value is the default for the Java VM.	service_max_native_size=131072
service_min_heap_size	Windows only. Specifies the minimum heap size for the adapter Java VM. Possible values are the valid Java VM heap sizes. The default value is the default Java VM heap size.	service_min_heap_size=536870912
service_max_heap_size	Windows only. Specifies the maximum heap size for the adapter Java VM. Possible values are any valid Java VM heap sizes. The default value is 536870912.	service_max_heap_size=536870912
service_num_vm_args	Windows only. The number of service_vm_arg<number> parameters specified. Possible values are the number of service_vm_arg<number> parameters. There is no default value.	service_num_vm_args=1
service_vm_arg<number>	Windows only. Specifies any additional arguments to the Java VM. For example, to get line numbers in any of the stack traces, set service_vm_arg1=java.compiler=NONE. If there is a list of arguments to specify, use multiple parameters as shown in the example by incrementing the last digit starting with 1. Be sure to set the service_num_vm_args correctly. Possible values are any valid Java VM arguments. There is no default value.	service_vm_arg1=java.compiler=NONE service_vm_arg2=oai.adapter=.aq

Table 2–7 Adapter.ini Parameters

Parameter	Description	Example
service_jdk_version	Windows only. The JDK version the adapter Java VM should use. The default value is 1.4.1.	service_jdk_version=1.4.1
service_jdk_dll	Windows only. The dll the adapter Java VM should use. The default value is jvm.dll.	service_jdk_dll=jvm.dll
corba_port_number	The CORBA port number on which the adapter CORBA service listens. Generally, this port is allocated dynamically. However, it can be configured to enable access across firewall.	corba_port_number=14000
encoding	Character encoding for published messages. The adapter uses this parameter to generate encoding information in encoding tag of transformed OracleAS InterConnect message. OracleAS InterConnect represents messages internally as an XML document. The default encoding of the XML document is UTF-8. However, this encoding can be configured using this parameter, which is typically used when the OracleAS InterConnect message consists of characters not supported by UTF-8 and when the XMLParser is unable to handle them.	encoding=JA16SJIS

Table 2-7 Adapter.ini Parameters

Parameter	Description	Example
nls_date_format	Format for date fields expressed as string. The default date format is <code>EEE MMM dd HH:mm:ss zzz yyyy</code> . For the meaning of this string, see the list of reserved characters in Table 2-8	Date format pattern <code>dd/MMM/yyyy</code> can represent <code>01/01/2003</code> . <code>nls_date_format=dd-MMM-yy</code> Multiple date formats can be specified as <code>num_nls_formats=2</code> <code>nls_date_format1=dd-MMM-yy</code> <code>nls_date_format2=dd/MMM/yy</code>
nls_country	This parameter is a valid ISO Country Code. These upper-case and two-letter codes are defined by ISO-3166. You can find a full list of these codes at a Web site, such as, http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html The default Country code is US. Note: This parameter specifies date format. It is applicable for the date format only.	US
nls_language	This parameter is a valid ISO Language Code. These lower-case and two-letter codes are defined by ISO-639. You can find a full list of these codes at a Web site, such as, http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt The default language code is en. Note: This parameter specifies date format. It is applicable for the date format only.	<code>nls_language=en</code>

[Table 2–8](#) shows the reserved characters used to specify the value of the `nls_date_format` parameter. Using these characters, you can construct a pattern 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; 96
M	Month in year	July; Jul; 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; 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 Parameters

Table 2–9 lists the parameters specific to the Advanced Queuing adapter.

Table 2–9 Advanced Queuing Adapter-specific Parameters

Parameter	Description	Example
<code>bridge_class</code>	This indicates the entry class for the Advanced Queuing adapter. Do not modify this value. A possible value is <code>oracle.oai.agent.adapter.aq.XMLAQBridge</code> . There is no default value.	<code>bridge_class=oracle.oai.agent.adapter.aq.XMLAQBridge</code>
<code>aq_bridge_username</code>	The schema username the bridge should connect to which dequeues or enqueues messages from a queue in order to publish or subscribe to events defined using iStudio. A possible value is <code>aquser</code> . There is no default value.	<code>aq_bridge_username=aquser</code>
<code>aq_bridge_password</code>	The schema password corresponding to the <code>aq_bridge_username</code> . A possible value is <code>aquser</code> . There is no default value.	<code>aq_bridge_password=welcome</code>
<code>aq_bridge_thin_jdbc</code>	This indicates whether to use thin JDBC when talking to the database. The possible values are <code>true</code> and <code>false</code> . The default is <code>true</code> .	<code>aq_bridge_thin_jdbc=true</code>
<code>aq_bridge_host</code>	Name of the machine hosting the database instance specified by <code>aq_bridge_instance</code> . The possible values are the name of the machine and there is no default value.	<code>aq_bridge_host=mpjoshi-pc</code>
<code>aq_bridge_instance</code>	The SID of the database instance. The possible value is the valid SID and there is no default value.	<code>aq_bridge_instance=orcl</code>
<code>aq_bridge_port</code>	The port on which the TNS listener is running for the database instance specified by <code>aq_bridge_instance</code> . The possible value is a valid TNS listener number and there is no default value.	<code>aq_bridge_port=1521</code>
<code>aq_bridge_consumer_name</code>	If all the queues that this adapter will connect to on the application database side are single consumer queues, this can be left blank. If, however, any of the queues is a multiconsumer queue, then specify a consumer name. The possible value is the same as the <code>aq_bridge_username</code> . There is no default value.	<code>aq_bridge_consumer_name=aquser</code>
<code>aq_bridge_owner</code>	The owner of the advanced queue. The possible value is the same as <code>aq_bridge_username</code> and there is no default value.	<code>aq_bridge_owner=aquser</code>

RAC-specific adapter.ini Parameters for the Advanced Queuing Adapter

When the Advanced Queuing adapter is servicing a RAC database as the spoke database, parameters listed in [Table 2–10](#) represent information on connection and configuration.

Table 2–10 RAC-specific adapter.ini Parameters

Parameter	Description	Example
aq_bridge_num_nodes	Indicates the number of nodes in RAC cluster.	aq_bridge_num_nodes=4
aq_bridge_hostx	Indicates host for the node x.	aq_bridge_host2=dsunram13
aq_bridge_port	Indicates port for node x.	aq_bridge_port2=1421
aq_bridge_instancex	Indicates instance on node x.	aq_bridge_instance2=orcl2

Starting the Advanced Queuing Adapter

On UNIX, start the Advanced Queuing adapter using the `start` script in the following directory:

```
$ORACLE_HOME/oai/9.0.4/adapters/Application
```

Type **start**, then press **Enter**.

On Windows, start the Advanced Queuing adapter from the Services window available from the Start menu.

1. Access the Services window from the Start menu:

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

The Services window appears.

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

On...	Choose...
Windows NT	Choose Start.
Windows 2000	Right click the service and choose Start from the menu.

Stopping the Advanced Queuing Adapter

On UNIX, stop the Advanced Queuing adapter using the `stop` script in the following directory:

```
$ORACLE_HOME/oai/9.0.4/adapters/Application
```

Type **stop**, then press **Enter**.

On Windows, stop the adapter from the Services window available from the Start menu.

1. Access the Services window from the Start menu:

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

The Services window appears.

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

On...	Choose...
Windows NT	Choose Stop.
Windows 2000	Right click the service and choose Stop from the menu.

You may verify the stop status by viewing the `oai_log.txt` files in the appropriate time stamped subdirectory of the `log` directory within the adapter directory.

Design Time and Runtime Concepts

This chapter describes the design time and runtime concepts for the Advanced Queuing adapter. Topics discussed are:

- [Advanced Queuing Adapter Design Time Concepts](#)
- [Advanced Queuing Adapter Runtime Concepts](#)
- [Starting the Advanced Queuing Adapter](#)
- [Stopping the Advanced Queuing Adapter](#)

Advanced Queuing Adapter Design Time Concepts

The following topics discuss the iStudio concepts pertinent to the Advanced Queuing adapter.

Supported Features

The Advanced Queuing adapter can handle the following payload types:

- RAW with XML data.
- Oracle Object (can be hierarchical) where any field can contain XML data.

RAW Payload

Using iStudio, a DTD can be imported and an application can be configured where the corresponding XML message can be picked up or placed in by the OracleAS InterConnect adapter. If the queue has been configured for RAW payload, the message payload is plain XML.

See Also: [Chapter 4, "Sample Use Cases"](#)

Oracle Object Payload with and without XML Data

In addition to XML, the Advanced Queuing adapter supports Oracle Object Types. The Advanced Queuing adapter provides complete flexibility to import the Advanced Queue's payload Oracle Object Type. Therefore, any, some, or all of the attributes can be associated within this Oracle Object Type to be of different XML types.

For example, to send a `Customer` and `PurchaseOrder` as part of one OracleAS InterConnect message, the corresponding DTDs are contained in the `Customer.DTD` and `PurchaseOrder.DTD` files. 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 enqueueing 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 in order to complete the example:

1. Create iStudio datatypes corresponding to the XML DTDs by importing them.

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. This is a workaround for a known iStudio issue.

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 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`. The remainder of the process is the same as for other messages.

Creating Metadata in iStudio

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

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

Importing from XML:

1. Select Import XML on the Publish or Subscribe Wizard.
2. In the File dialog, select the DTD file.
3. From the list of all nodes, select root element.

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

The following are salient points when working with Advanced Queuing adapter:

- Common view—Create by importing from XML and specifying the appropriate DTD file.
- Application view:
 - Raw Payload—Import from XML.
 - Object Payload—Import from the database by select the appropriate queue payload.
- Always specify the Message Type as AQ for Advanced Queue applications.
- Event map usage.
- Follow these steps to specify the application queues:
 1. On the deploy tab in iStudio, expand the Applications container.
 2. Expand applicationName.
 3. Expand the Routing container.
 4. Right Click on Application Queues and select Edit.
 5. Enter the Application Queue Names for the Advanced Queuing adapter to subscribe and publish messages to.
 6. Click OK.

Returned In Arguments

Returned In Arguments are used only when invoking procedures. The concept of Returned In Arguments is to propagate INOUT attributes contained in the request to the reply as well. Without this feature, these attributes would have to exist in both the common view and the application view of the implementor and are INOUT. All these mappings would have to be one to copy these attributes on their way out and back in, for example, when receiving the reply. Therefore, the user

could use one of these returned in arguments to correlate the reply with an asynchronous request.

For example, 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 `INOUT`, for example, both in the request and the reply, then it should be an `INOUT` parameter. Click `Returned In Args` in the `Invoke Wizard` and select `ID` in the `Please Select In Arguments` and the `Please Select Out Arguments`.

Advanced Queuing Adapter Runtime Concepts

The following section describes the runtime concepts of the Advanced Queuing adapter.

How the Advanced Queuing Adapter Works

The following topics describe how the Advanced Queuing adapter works.

Sending Adapter

The Advanced Queuing adapter is comprised of the Advanced Queuing 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 InterConnect message waiting to be sent by the adapter. The adapter then picks up the message, builds the corresponding OracleAS InterConnect message, persists it, transforms it to the common view, and routes it to the hub. From the hub, the message is routed to the appropriate subscriber based on configuration done using iStudio which could be content based or subscription based.

The application and the Advanced Queuing adapter communicate via the publishing and invoking queues, residing in the `aq_bridge_username` parameter, for outbound messages and via subscribing and implementing queues for inbound messages. Therefore, the Advanced Queuing adapter is down while the application is publishing OracleAS InterConnect messages. These messages are held in the queues and will be picked up in the order they were enqueued by the Advanced Queuing adapter once it is up and running. If there are messages in the queues which should no longer be published, dequeue them manually.

Receiving Adapter

On the subscribing or receiving side, the Advanced Queuing 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 on the Deploy tab of iStudio. The application should pick this message up from this. If the Advanced Queuing adapter were an implementor instead of a subscriber, 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.

Starting the Advanced Queuing Adapter

Start the Advanced Queuing adapter using the `start` script in the directory named after the Advanced Queuing adapter. On Windows or Windows 2000, start it from the Service window available from the Start menu.

1. Access the Services window from the Start menu:

Platform	Action
Windows 2000	Start > Settings > Control Panel > Administrative Tools > Services

The Services window displays.

2. Select the *OracleHomeOracleASInterConnectAdapter-Application* service.

3. Start the service based on your operating system:

Platform	Action
Windows 2000	Right click the service and choose Start from the menu that displays.

See Also: ["Configuring the Advanced Queuing Adapter"](#) on page 3-2 for the location of the `start` script

Sample Log File of Successfully Started Advanced Queuing Adapter

Startup status can be verified by viewing the `oailog.txt` files in the appropriate timestamped subdirectory of the `log` directory of the Advanced Queuing adapter directory. Subdirectory names take the following form:

```
timestamp_in_milliseconds
```

The following file displays information about an Advanced Queuing adapter that started successfully:

```
D:\oracle\ora904\oai\9.0.4\adapters\aqapp>D:\oracle\ora904\oai\9.0.4\bin\Java
Service.exe -debug "Oracle OAI Adapter 9.0.4 - aqapp"
D:\oracle\ora904\oai\9.0.4\adapters\aqapp adapter.ini
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.
```

Stopping the Advanced Queuing Adapter

Stop the Advanced Queuing adapter using the `stop` script in the directory named after the Advanced Queuing adapter. On Windows 2000, stop the adapter from the Services window available from the Start menu.

1. Access the Services window from the Start menu:

Platform	Action
Windows 2000	Start > Settings > Control Panel > Administrative Tools > Services

The Services window displays.

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

Platform	Action
Windows 2000	Right click the service and choose Stop from the menu that displays.

Stop status can be verified by viewing the `oailog.txt` files in the appropriate timestamped subdirectory of the log directory of the adapter directory.

See Also: ["Configuring the Advanced Queuing Adapter"](#) on page 3-2 for the location of the `start` script

4

Sample Use Cases

This chapter describes sample use cases for the Advanced Queuing adapter.

Advanced Queuing Adapter Sample Use Cases

The following are sample use cases for the Advanced Queuing adapter.

Case One: The Advanced Queuing Publishes and Subscribes with RAW Payload

The Advanced Queuing adapter publishes and subscribes using RAW Payload. The following case uses one installation of an Advanced Queuing adapter that publishes a message and subscribes to the same event. Therefore, the log files will indicate that an outbound message is followed by an inbound message.

Using SQL*PLUS utility, 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;
```

Design Time Steps

The following steps describe the procedures necessary to publish and subscribe a message containing RAW XML payload. The event created is `createCustomer` where the payload corresponds to the `customer.dtd` provided. For outbound messages, the application queues are `xml_raw_q1`. For inbound messages, the application queues are `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 tree 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 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.
- 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 tree 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.

Runtime Steps:

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 in SQL*PLUS. Log in as the system user.
2. Log in as `aquser/manager` using the SQL*PLUS utility.

3. Create Advanced Queues by executing the `CreateAQ.sql` script.
See Also: ["Case Two: The Advanced Queuing Invokes and Implements with Oracle Object Type Payload"](#) on page 4-5
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 Advanced Queuing adapter.

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;
/

```

- **create_user.sql**

```

CREATE USER aquser IDENTIFIED BY manager;
GRANT CONNECT, RESOURCE TO aquser;
GRANT AQ_USER_ROLE, AQ_ADMINISTRATOR_ROLE TO aquser;

```

Case Two: The Advanced Queuing Invokes and Implements with Oracle Object Type Payload

The Advanced Queuing adapter invokes and implements with Oracle Object Type Payload. This case uses one installation of an Advanced Queuing adapter that sends a request, receives the request, sends back a reply, and receives the reply. Therefore, the log files will indicate four different message entries.

Design Time Steps

The following steps describe the procedures necessary to invoke and implement a procedure. An Oracle Object Type is used as a payload.

1. Log in as the `aquser/manager` using the SQL*PLUS utility.
2. Create the `addr` and `cust` Oracle Object Types by executing the `CreateADT.sql` file.
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. Create an application data type for the `aqapp` application. 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. Create a new invoked procedure for the `aqapp` application. 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 tree 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 tree and drill down to 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 aqapp: 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

Runtime Steps

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

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

See Also: ["Case Two: The Advanced Queuing Invokes and Implements with Oracle Object Type Payload"](#) on page 4-5
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`.

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;
/
```

Frequently Asked Questions

This chapter provides answers to frequently asked questions about the Advanced Queuing adapter. This chapter discusses the following topics:

- [Installation Questions](#)
- [Design Time Questions](#)

Installation Questions

The following questions address the installation of the Advanced Queuing adapter.

How do I know the Advanced Queuing adapter is started properly?

View the `oai.txt` file located in the appropriate timestamped subdirectory of the Advanced Queuing adapter log directory:

Platform	Directory
UNIX	<code>ORACLE_HOME/oai/9.0.4/adapters/Application/log/timestamp_in_milliseconds</code>
Windows	<code>ORACLE_HOME\oai\9.0.4\adapters\Application\log\timestamp_in_milliseconds</code>

If there are no exceptions, the Advanced Queuing adapter has started properly.

The Advanced Queuing adapter did not start properly—what went wrong?

View the exceptions in the Advanced Queuing adapter log file (`oai.log.txt`). The exceptions should provide some idea about what went wrong. It is possible that the Advanced Queuing adapter is unable to connect to the repository. Make sure the repository is started properly. The Advanced Queuing adapter will connect to the Repository once it is started properly. You do not need to restart the Adapter.

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

Why is the Advanced Queuing adapter using old information after I changed information in iStudio?

The Advanced Queuing adapter caches the information from iStudio (the information which is stored in the Repository) locally for better performance in a production environment.

If you change something in iStudio and want to see it in the runtime environment, stop the Advanced Queuing 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 should allow the adapter to obtain the new metadata from the repository when started.

Which databases are referred to during installation?

The database the questions are referring to is the database on the application side from which the adapter will either put or get messages from Advanced Queuing.

What is the consumer name?

If all the queues the Advanced Queuing adapter connects to on the application database side are single consumer queues, 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 Advanced Queuing adapter uses a consumer name to indicate to OracleAS InterConnect to pick up this message. The following two options help you to find out the consumer name to use:

- If the piece of code that writes the message to the Advanced Queuing adapter is already written, 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 Advanced Queuing adapter is not written, type in any string as the consumer name. When that piece of code is built, ensure that the consumer names match.

Can I edit configuration settings after installation?

Yes, edit the parameters in the following file:

Platform	Directory
UNIX	<code>ORACLE_HOME/oai/9.0.4/adapters/Application/adapter.ini</code>
Windows	<code>ORACLE_HOME\oai\9.0.4\adapters\Application\adapter.ini</code>

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

Parameter	Parameter Information
<code>aq_bridge_username</code>	username
<code>aq_bridge_password</code>	Password
<code>aq_bridge_host</code>	Host
<code>aq_bridge_port</code>	The TNS listener port.
<code>aq_bridge_instance</code>	The database SID.

Parameter	Parameter Information
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_consumer_name	The consumer name.
aq_bridge_thinjdbc	Use a THIN JDBC driver if true, otherwise use OCI8 JDBC driver.

Design Time Questions

The following questions address design time concepts for the Advanced Queuing adapter.

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

The installer overwrites previous installations of the Advanced Queuing 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 Advanced Queuing adapter in each Oracle home. When you install the Advanced Queuing adapter a second time, choose a different Oracle home from the first Advanced Queuing adapter.

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

ANY tags in an XML DTD allow unstructured data in XML to be used. OracleAS 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 OracleAS InterConnect to know about the structure:

1. The simplest method is to modify the DTD being importing into iStudio and replace the ANY tag with structured data. When modifying the DTD, only a copy of the DTD being importing 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 a 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 only has zip, a standard DTD union mechanism for doing this can be used.

2. The following steps describe a second approach which involves creating a separate DTD which defines the structure used at runtime for the ANY tag.
 - a. Import the DTD for the event, either while creating an ADT or while creating the published or subscribed event or the invoked or 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 a 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>
```

Then import the address_any.dtd by right-clicking on the address ADT in iStudio. This assumes 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>
```

Index

A

advanced queuing adapter
 configuration, 2-5
 database requirements, 1-3
 design time concepts, 3-2
 hardware requirements, 1-2
 how it works, 3-5
 installation, 2-2
 jre requirements, 1-3
 operating system requirements, 1-2
 overview, 1-2
 runtime concepts, 3-5
 sample use cases, 4-2
 software requirements, 1-2
 starting, 3-6
 stopping, 3-8
application parameter, 2-6

C

configuration, 2-5
 adapter.ini, 2-9
 advanced queuing adapter parameters, 2-17
 directories, 2-6
 hub.ini, 2-7
 ini file settings, 2-7

D

database
 requirements, 1-3
design time
 concepts, 3-2

metadata, 3-3
questions, 5-4
supported features, 3-2

F

frequently asked questions
 design time, 5-4
 installation, 5-2

I

installation, 2-2
 questions, 5-2
 tasks, 2-3
iStudio
 creating metadata, 3-3

J

jre
 requirements, 1-3

M

metadata
 in istudio, 3-3

P

preinstallation
 tasks, 2-2

R

RAC-specific adapter.ini parameters, 2-18
receiving adapter, 3-6
runtime
 concepts, 3-5

S

sample use cases, 4-2
 invoke and implements with adt payload, 4-5
 publish and subscribe with raw payload, 4-2
sender adapter, 3-5
supported features, 3-2
 oracle object payload with and without xml
 data, 3-2
 raw payload, 3-2
 returned in arguments, 3-4