# Oracle9*i*AS InterConnect Adapter for AQ

Installation and User's Guide

Release 2 (9.0.2)

February 2002

Part No. A95449-01

ORACLE

Oracle9*i*AS InterConnect Adapter for AQ Installation and User's Guide, Release 2 (9.0.2)

Part No. A95449-01



Primary Author: Maneesh Joshi

# Contents

## Index

# Send Us Your Comments

**Oracle9*i*AS InterConnect Adapter for AQ Installation and User's Guide, Release 2 (9.0.2)**

**Part No. A95449-01**

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 - iasdocs_us@oracle.com
- Fax - (650) 506-7407 Attn: Oracle9*i* Application Server Documentation Manager
- Postal service:

  Oracle Corporation
  Oracle9*i* Application Server Documentation Manager
  500 Oracle Parkway, M/S 2op3
  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

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.

## Organization

This document contains:

### Chapter 1, "Introduction"

This chapter describes the 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:

- Oracle9iAS InterConnect User's Guide in the Oracle9i Application Server Documentation Library
- Oracle9iAS InterConnect Adapter Configuration Editor User's Guide

In North America, printed documentation is available for sale in the Oracle Store at

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

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

`http://www.oraclebookshop.com/`

Other customers can contact their Oracle representative to purchase printed documentation.

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/admin/account/membership.html`

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/docs/index.htm`

# 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**. |
| *Italics* | Italic typeface indicates book titles or emphasis. | *Oracle9i Database Concepts*<br>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, 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.<br>You can back up the database by using the `BACKUP` command.<br>Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view.<br>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.<br><br>**Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown. | Enter `sqlplus` to open SQL*Plus.<br><br>The password is specified in the `orapwd` file.<br><br>Back up the datafiles and control files in the `/disk1/oracle/dbs` directory.<br><br>The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table.<br><br>Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`.<br><br>Connect as `oe` user.<br><br>The `JRepUtil` class implements these methods. |
| *`lowercase monospace (fixed-width font) italic`* | Lowercase monospace italic font represents placeholders or variables. | You can specify the *`parallel_clause`*.<br><br>Run `U`*`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 |
|---|---|---|
| [ ] | Brackets enclose one or more optional items. Do not enter the brackets. | `DECIMAL (`*`digits`*` [ , `*`precision`*` ])` |
| { } | Braces enclose two or more items, one of which is required. Do not enter the braces. | `{ENABLE \| DISABLE}` |
| \| | 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. | `{ENABLE \| DISABLE}`<br>`[COMPRESS \| NOCOMPRESS]` |

| Convention | Meaning | Example |
|---|---|---|
| ... | Horizontal ellipsis points indicate either:<br>■ That we have omitted parts of the code that are not directly related to the example<br>■ That you can repeat a portion of the code | `CREATE TABLE ... AS subquery;`<br>`SELECT col1, col2, ... , coln FROM employees;` |
| .<br>.<br>. | 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. | `acctbal NUMBER(11,2);`<br>`acct CONSTANT NUMBER(4) := 3;` |
| *Italics* | Italicized text indicates placeholders or variables for which you must supply particular values. | `CONNECT SYSTEM/system_password`<br>`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. However, because these terms are not case sensitive, you can enter them in lowercase. | `SELECT last_name, employee_id FROM employees;`<br>`SELECT * FROM USER_TABLES;`<br>`DROP TABLE hr.employees;` |
| lowercase | Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.<br>**Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown. | `SELECT last_name, employee_id FROM employees;`<br>`sqlplus hr/hr`<br>`CREATE USER mjones IDENTIFIED BY ty3MU9;` |

## 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 - *HOME_NAME* > 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 *command prompt* in this manual. | `C:\oracle\oradata>` |
| | 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 scott/tiger TABLES=emp QUERY=\"WHERE job='SALESMAN' and sal<1600\"`<br>`C:\>imp SYSTEM/`*`password`*<br>`FROMUSER=scott TABLES=(emp, dept)` |
| *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` |

| Convention | Meaning | Example |
|---|---|---|
| *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 that by default used one of the following names:<br>■ `C:\orant` for Windows NT<br>■ `C:\orawin95` for Windows 95<br>■ `C:\orawin98` for Windows 98<br>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`. If you install Oracle9*i* 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 *ORACLE_BASE*.<br>All directory path examples in this guide follow OFA conventions.<br>Refer to *Oracle9i Database Getting Starting 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 introduction to the Advanced Queuing adapter and how it works with Oracle9*i*AS InterConnect. This chapter discusses the following topics:

- Advanced Queuing Adapter Overview

# Advanced Queuing Adapter Overview

The Oracle9*i*AS InterConnect Advanced Queuing adapter enables an Oracle Advanced Queuing application to be integrated with other applications using Oracle9*i*AS 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 NT/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 NT/2000 | version that supports at least 400 MHz |
| IBM AIX-Based Systems | version 4.3.3 |
| Compaq Tru64 UNIX | version 5.0a or 5.1 |

| Operating System | Version |
|---|---|
| HP 9000 Series HP-UX | version 11.0 |
| SUSE LINUX SLES7 | version 7.2, Kernel 2.4.7, GilbC 2.2.2-55 |
| Sun SPARC Solaris | version 2.6 |

### JRE Requirements

Oracle9*i*AS InterConnect uses JRE 1.3.1 which is installed with its components.

### Database Requirements

The Oracle Database 8*i* and later Oracle Databases 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.

# 2

# 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
- Advanced Queuing Adapter Configuration

# Installing the Advanced Queuing Adapter

This section describes the following topics:



## Preinstallation Tasks

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

- An existing Oracle9*i* Application Server Oracle home
- An existing Oracle9*i* Application Server Infrastructure Database Oracle home
- An existing Oracle9*i*AS InterConnect Oracle home
- A new Oracle home (the installer creates this for you)

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

- *Oracle9i Application Server Installation Guide*, which includes information on:
  - CD-ROM mounting
  - Oracle Universal Installer startup
  - Oracle9iAS InterConnect installation
- *Oracle9iAS InterConnect User Guide*, which includes information on:
  - Oracle9iAS InterConnect software, hardware, and system requirements

---

**Note:** Oracle9*i*AS InterConnect Hub is installable through the Oracle9*i*AS InterConnect Hub installation type. You must install the Oracle9*i*AS 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 Oracle9*i*AS InterConnect installation, select Advanced Queuing adapter, then select Next.

   Consider the following scenarios:

   - If installing in a Oracle9*i*AS Oracle home, make sure the Oracle9*i*AS InterConnect hub has been installed in this Oracle home. Continue to step three.
   - If installing the Advanced Queuing adapter in the Oracle home that does not contain Oracle9*i*AS, make sure the Oracle9*i*AS InterConnect Hub is already installed. Continue to step two.

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

2. If installing Oracle9*i*AS 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 Oracle9*i*AS 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 Oracle Applications 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 | `%ORACLE_HOME%\oai\9.0.2\adapters\`*`Application`* |
| UNIX | `$ORACLE_HOME/oai/9.0.2/adapters/`*`Application`* |

# Advanced Queuing Adapter Configuration

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 | `$ORACLE_HOME/oai/9.0.2/adapters/`*`Application`* |
| Windows | `%ORACLE_HOME%\oai\9.0.2\adapters\`*`Application`* |

***Table 2–2 Executable Files***

| File | Description |
|---|---|
| `start.bat` (Windows) | Takes no parameters, starts the adapter. |
| `start` (UNIX) | |
| `stop.bat` (Windows) | Takes no parameters; stops the adapter. |
| `stop` (UNIX) | |

| File | Description |
|---|---|
| `ignoreErrors.bat` (Windows)<br>`ignoreErrors` (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 `ErrorCodes.ini` will be ignored. |

***Table 2–3 Configuration Files***

| File | Description |
|---|---|
| `ErrorCodes.ini` (Windows and UNIX) | Should contain one error code per line. |
| `adapter.ini` (Windows and UNIX) | Consists of all the initialization parameters which the adapter reads at startup. Refer to Appendix A for a typical `adapter.ini` file. |

***Table 2–4 Directories***

| File | Description |
|---|---|
| persistence | The messages are persisted in this directory. This directory or its contents should not edited |
| logs | 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 `oailog.txt` 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 *i*Studio under the Applications folder.

If you are using pre-packaged metadata, after importing the pre-packaged metadata into the repository, start up *i*Studio to find the corresponding application (under the Applications folder in *i*Studio) 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

The following are `.ini` files are used in configuring the Advanced Queuing adapter.

### Hub.ini

The Advanced Queuing adapter connects to the hub database using parameters from the `hub.ini` file located in the hub directory. The following table lists the parameter name, a description for each parameter, the possible and default values, and an example.

| Parameter | Description | Example |
|---|---|---|
| `hub_username` | The name of the hub database schema (or username). Possible values are valid hub database username. There is no default value. | `hub_username=myhub` |
| `hub_password` | The password for the hub database user. Possible values are the valid password for the hub database user. There is no default value. | `hub_password=manager` |
| `hub_host` | The name of the machine hosting the hub database. Possible values are the valid machine name. There is no default value. | `hub_host=mpjoshipc` |
| `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. There is no default value. | `repository_name=myrepo` |

### Adapter.ini

The Advanced Queuing adapter connects to the spoke application using parameters from the `adapter.ini` file. The following table lists the parameter name, a description for each parameter, the possible and default values and an example.

| Parameter | Description | Example |
|---|---|---|
| `application` | 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. | `application=aqapp` |
| `partition` | The partition this adapter handles as specified in iStudio. Any alphanumeric string is a possible value. There is no default value. | `partition=germany` |
| `instance_number` | 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 1. There is no default value. | `instance_number=1` |
| `agent_log_level` | Specifies the amount of logging necessary. Possible values are:<br>0=errors only<br>1=status and errors<br>2=trace, status, and errors<br>The default value is 1. | `agent_log_level=2` |
| `agent_ subscriber_name` | 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. | `agent_subscriber_ name=aqapp` |
| `agent_message_ selector` | 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. | `agent_message_ selector=recipient_ list like '%aqapp,%'` |
| `agent_reply_ subscriber_name` | 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: application`) concatenated with instance number (`parameter: instance_number`). There is no default value. | `If application=aqapp, instance_number=2, then, agent_reply_ subscriber_name=aqapp2` |

| 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=receipient_ 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 Oracle9*i*AS InterConnect-specific DTD for all messages sent to the hub as other Oracle9*i*AS 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 Oracle9*i*AS InterConnect DTD. Only set to true if a Oracle9*i*AS 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:<br>■ startup—Cache everything at startup. This may take a while if there are a lot of tables in the repository.<br>■ demand—Cache metadata as it is used.<br>■ none—No caching. This slows down performance.<br>The default value is demand. | agent_metadata_ caching=demand |

| Parameter | Description | Example |
|---|---|---|
| agent_dvm_table_caching | Specifies the DVM caching algorithm. Possible values are:<br>■ startup—Cache all DVM tables at startup. This may take a while if there are a lot of tables in the repository.<br>■ demand—Cache tables as they are used.<br>■ none—No caching. This slows down performance.<br>The default value is demand. | agent_dvm_table_caching=demand |
| agent_lookup_table_caching | Specifies the lookup table caching algorithm. Possible values are:<br>■ startup—Cache all lookup tables at startup. This may take a while if there are a lot of tables in the repository.<br>■ demand—Cache tables as they are used.<br>■ none—No caching. This slows down performance.<br>The default value demand. | agent_lookup_table_caching=demand |
| agent_delete_file_cache_at_startup | With any of the agent caching methods enabled, metadata from the repository is cached locally on the file system.<br>Set this parameter to true to delete all cached metadata on startup.<br>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.<br>Possible values are true or false. The default value is false. | agent_delete_file_cache_at_startup=false |
| agent_max_ao_cache_size | Specifies the maximum number of application objects' metadata to cache. Possible values are any integer greater than 1. The default value is 200. | agent_max_ao_cache_size=200 |
| agent_max_co_cache_size | Specifies the maximum number of common objects' metadata to cache. Possible values are any integer greater than 1. The default value is 100. | agent_max_co_cache_size=100 |
| agent_max_message_metadata_cache_size | Specifies the maximum number of messages' metadata to cache (publish/subscribe and invoke/implement). Possible values are any integer greater than 1. The default value is 200. | agent_max_message_metadata_cache_size=200 |

| 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 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 1. The default value is 200. | agent_max_lookup_table_cache_size=200 |
| agent_max_queue_size | Specifies the maximum size that internal Oracle9*i*AS InterConnect message queues can grow. Possible values are any integer greater than 1. The default value is 1000. | agent_max_queue_size=1000 |
| agent_persistence_queue_size | Specifies the maximum size that internal Oracle9*i*AS InterConnect persistence queues can grow. Possible values are any integer greater than 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 30000. 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 Oracle9*i*AS InterConnect message. Possible values are any integer greater than 5000. The default value is 60000. | agent_persistence_retry_interval=60000 |
| 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\ora902\bin |
| 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\ora902\oai\902\lib\oai.jar;%JREHOME%\lib\i18n.jar;D:\oracle\ora902\jdbc\classes12.zip |
| service_class | The entry class for the Windows NT service. The 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 |

| Parameter | Description | Example |
|---|---|---|
| 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<br>service_vm_arg2=oai.adapter=.aq |
| service_jdk_version | Windows only. The JDK version the adapter Java VM should use. The default value is 1.3.1. | service_jdk_version=1.3.1 |
| service_jdk_dll | Windows only. The dll the adapter Java VM should use. The default value is jvm.dll. | service_jdk_dll=jvm.dll |

### Advanced Queuing Adapter Parameters

The following table lists the parameters specific to the Advanced Queuing adapter.

| Parameter | Description | Example |
|---|---|---|
| bridge_class | This indicates the entry class for the Advanced Queuing adapter. Do not modify this value. A possible value is oracle.oai.agent.adapter.aq. XMLAQBridge. There is no default value. | bridge_ class=oracle.oai.agent. adapter.aq. XMLAQBridge |
| aq_bridge_ username | The schema username the bridge should connect to which dequeus or enqueues messages from a queue in order to publish or subscribe to events defined using iStudio. A possible value is aquser. There is no default value. | aq_bridge_ username=aquser |
| aq_bridge_ password | The schema password corresponding to the aq_ bridge_username. A possible value is aquser. There is no default value. | aq_bridge_ password=welcome |
| aq_bridge_thin_ jdbc | This indicates whether to use thin JDBC when talking to the database. The possible values are true and false. The default is true. | aq_bridge_thin_ jdbc=true |
| aq_bridge_host | Name of the machine hosting the database instance specified by aq_bridge_instance. The possible values are the name of the machine and there is no default value. | aq_bridge_ host=mpjoshi-pc |
| aq_bridge_ instance | The SID of the database instance. The possible value is the valid SID and there is no default value. | aq_bridge_instance=orcl |
| aq_bridge_port | The port on which the TNS listener is running for the database instance specified by aq_bridge_instance. The possible value is a valid TNS listener number and there is no default value. | aq_bridge_port=1521 |
| aq_bridge_ consumer_name | 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 aq_bridge_ username. There is no default value. | aq_bridge_consumer_ name=aquser |
| aq_bridge_owner | The owner of the advanced queue. The possible value is the same as aq_bridge_username and there is no default value. | aq_bridge_owner=aquser |

# 3

# Design Time and Runtime Concepts

This chapter describes the design time and runtime concepts for 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 Oracle9*i*AS 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 Oracle9*i*AS 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 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 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 Oracle9*i*AS 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:** *Oracle9iAS InterConnect User 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:** *Oracle9iAS InterConnect User 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 Oracle9*i*AS InterConnect message waiting to be sent by the adapter. The adapter then picks up the message, builds the corresponding Oracle9*i*AS 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 Oracle9*i*AS 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:

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

   The Services window displays.

2. Select the *OracleHome9iASInterConnectAdapter-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 that displays. |

**See Also:** "Advanced Queuing Adapter Configuration" 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\ora902\oai\9.0.2\adapters\aqapp>D:\oracle\ora902\oai\9.0.b•in\Java
Service.exe -debug "Oracle OAI Adapter 9.0.2 - aqapp"
D:\oracle\ora9021\oai\9.0.2\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 NT or Windows 2000, 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 displays.

2. Select the *OracleHome9iASInterConnectAdapter-Application* service.
3. Start 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 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:** "Advanced Queuing Adapter Configuration" 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. This chapter discusses the following topics:

- Advanced Queuing Adapter Sample Use Cases

# 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;
/
```

# 5

# 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 | `$ORACLE_HOME/oai/9.0.2/adapters/`*`Application`*`/log/`*`timestamp_in_milliseconds`* |
| Windows | `%ORACLE_HOME%\oai\9.0.2\adapters\`*`Application`*`\log\`*`timestamp_in_milliseconds`* |

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 (`oailog.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:** *Oracle9iAS 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 Oracle9*i*AS 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 | `$ORACLE_HOME/oai/9.0.2/adapters/`*`Application`*`/adapter.ini` |
| Windows | `%ORACLE_HOME%\oai\9.0.2\adapters\`*`Application`*`\adapter.ini` |

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

| Parameter | Parameter Information |
|---|---|
| `aq_bridge_username` | username |
| `aq_bridge_password` | Password |
| `aq_bridge_host` | Host |
| `aq_bridge_port` | The TNS listener port. |
| `aq_bridge_instance` | 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. Oracle9*i*AS 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 Oracle9*i*AS 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

## R



## S