Oracle9*i*AS InterConnect Adapter for MQ Series

Installation and User's Guide

Release 2 (9.0.2)

February 2002

Part No. A95443-01

ORACLE

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

Part No. A95443-01

Copyright © 2002 Oracle Corporation. All rights reserved.

Primary Author: Bo Stern

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, Oracle9iAS Discoverer, SQL*Plus, and PL/SQL are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

2 Installation and Configuration

3 Design Time and Runtime Concepts

4 Frequently Asked Questions

Index

Send Us Your Comments

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

Part No. A95443-01

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 MQ Series adapter and the hardware and software requirements.

Chapter 2, "Installation and Configuration"

This chapter describes installation and configuration of the MQ Series adapter.

Chapter 3, "Design Time and Runtime Concepts"

This chapter describes the design time and runtime concepts for the MQ Series adapter.

Chapter 4, "Frequently Asked Questions"

This chapter provides answers to frequently asked questions about the MQ Series 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* 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. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width font)`	Lowercase monospace typeface indicates 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 `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. 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`*. 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}` `[COMPRESS | NOCOMPRESS]`
...	Horizontal ellipsis points indicate either:	
	■ That we have omitted parts of the code that are not directly related to the example	`CREATE TABLE ... AS subquery;`
	■ That you can repeat a portion of the code	`SELECT col1, col2, ... , coln FROM employees;`
. . .	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);` `acct CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/system_password` `DB_NAME = database_name`

Convention	Meaning	Example

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;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

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.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`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\"`
		`C:\>imp SYSTEM/`*`password`*
		`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: ■ `C:\orant` for Windows NT ■ `C:\orawin95` for Windows 95 ■ `C:\orawin98` for Windows 98 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*. All directory path examples in this guide follow OFA conventions. 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 discusses the following topic:

- MQ Series Adapter Overview

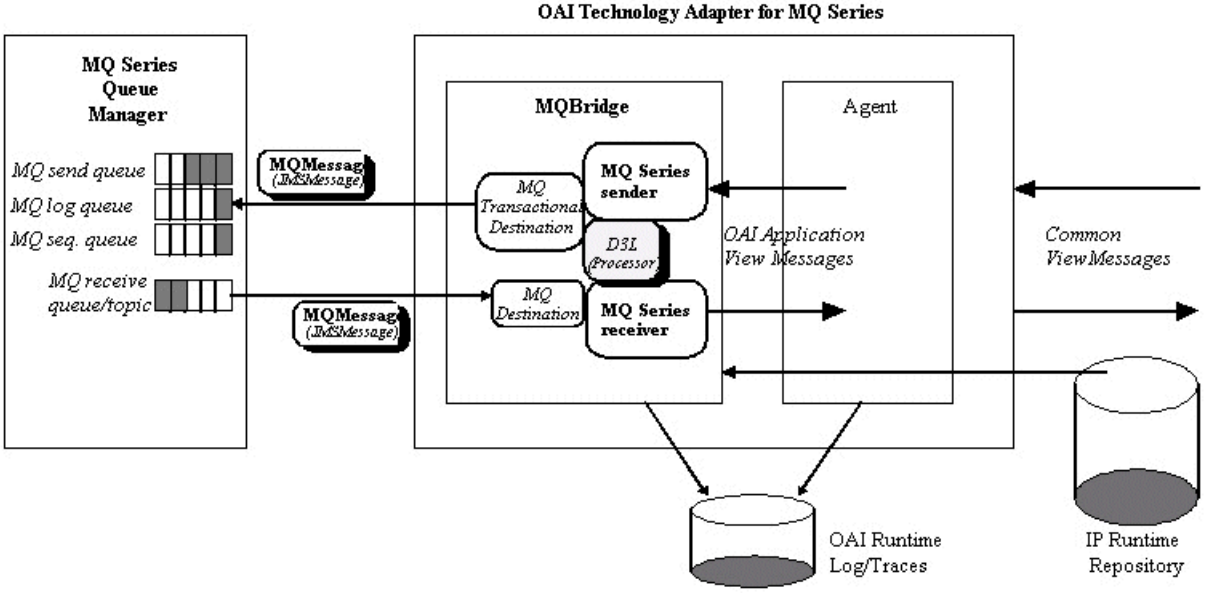
MQ Series Adapter Overview

The Oracle Technology Adapter for IBM MQ Series enables Oracle9*i*AS InterConnect to send messages to and receive messages from the MQ Series queues and topics. This allows an application that uses IBM's MQ Series as its messaging technology to be integrated with other applications using Oracle9*i*AS InterConnect. Therefore, the MQ Series adapter is useful in all enterprise application integration scenarios involving MQ Series-based applications.

This guide explains all necessary design time and runtime concepts of the MQ Series adapter.

The following diagram describes how the MQ Series adapter interacts with an MQ Series queue manager and Oracle9*i*AS InterConnect.

Figure 1–1 How the MQ Series Adapter Interacts with an MQ Series Queue Manager and Oracle9iAS InterConnect



The MQ Series adapter uses four logical queues or destinations to support its interaction with the MQ Series messaging system:

- Three for inbound messages; from Oracle9*i*AS InterConnect to MQ Series.
- One for outbound messages; from MQ Series to Oracle9*i*AS InterConnect.

The need for three queues for inbound messages is mandated by a requirement to support the sending of messages from Oracle9*i*AS InterConnect to MQ Series in a transactionally safe manner:

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

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

See Also: "Inbound" on page 1-7

Hardware Requirements

Table 1–1 lists the hardware requirements for the computer on which the Oracle MQ Series 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 sections list software requirements for the MQ Series adapter:

- Operating System Requirements
- JRE Requirements
- MQ Series Requirements

Operating System Requirements

Table 1–2 lists operating system requirements for the computer on which the MQ Series 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
HP 9000 Series HP-UX	version 11.0
SUSE LINUX SLES7	version 7.2, Kernel 2.4.7, GlibC 2.2.2-55
Sun SPARC Solaris	version 2.6 through 2.8

JRE Requirements

Oracle9*i*AS InterConnect uses JRE 1.3, which is installed with its components.

MQ Series Requirements

The following table lists the minimum software requirements for installing the MQ Series adapter.

Software	Supported Versions
MQ Series Server	Version 5.1 or 5.2
MQ Series Client[1]	Version 5.1 or 5.2

Software	Supported Versions
MQ Series classes for Java and MQ Series classes for Java Message Service (JMS)	Version 1.1.3 or latest supporting MQ Series 5.1 or 5.2 Downloadable from http://www-4.ibm.com/software/ts/mqseries/txppacs/ma88.html
MQ Series - Publish/Subscribe[2]	Version 1.0.6 or latest supporting MQ Series 5.1 or 5.2 Downloadable from http://www-4.ibm.com/software/ts/mqseries/txppacs/ma0c.html

[1] Only needed if using remote or non-local MQ Series queue manager.

[2] Only needed if topic (publish/subscribe) support in MQ Series is required.

Knowledge Requirements

The installation of the MQ Series adapter software components mentioned in "MQ Series Requirements" on page 1-4 should be performed by a MQ Series system administrator.

Configuring and using the MQ Series adapter requires the following:

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

The MQ Series queues and topics referred to in this guide must be created and started by the user before starting the MQ Series adapter.

MQ Series Adapter: Typical Use

The MQ Series adapter is primarily used to provide messaging capabilities between Oracle9*i*AS InterConnect and the MQ Series queueing systems, such as supporting the publish and subscribe paradigm for message exchanges, for example, sending or receiving orders, invoices, and product updates.

In addition to basic publish and subscribe messaging, the MQ Series adapter also supports the Oracle9*i*AS InterConnect request and reply paradigm which maps directly onto MQ Series' own generic support for request and reply messages. This capability is based on the support for message correlation for Oracle9*i*AS InterConnect as well as in MQ Series. Examples include inventory lookups, price calculations, and status requests.

Supported MQ Series Adapter Interfaces

The following sections describe the MQ Series adapter interfaces.

General

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

This format is derived from uniform resource identifiers and allows the specification of remote queues (queues on a queue manager other than the one to which you have connected) as well as the setting of the other queue connection properties.

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

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

Queue URI Example:

```
queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
```

Note: The name of the queue manager is omitted. This is interpreted to mean use the default queue manager as specified in the `adapter.ini` file.

The URI for a topic starts with the sequence `topic://` followed by the name of the topic, and optionally, a list of name-value pairs to set the remaining topic properties as follows:

```
topic://TopicName[?property=value [&property=value ]*]
```

Topic URI example:

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

> **Note:** The topic URI syntax does not allow the specification of the queue manager. It must be specified in the `mq.default.queue_manager` property in the `adapter.ini` file.

Inbound

Inbound interfaces consist of MQ Series queues to which the MQ Series adapter sends messages. The MQ Series adapter only supports MQ Series queues, not topics, for *inbound* interfaces. The reason for this stems from the following combination of constraints:

- The MQ Series adapter must send messages to MQ Series in a transactionally safe manner as it implements the Oracle9*i*AS InterConnect SDK TransactionalMessageReceiver interface.
- This requires the use of a queue for keeping log records.
- The destination queue or topic and log queue must be updated within the same JMS transaction, that is, the same JMS session.
- The MQ Series JMS implementation does not support the concept of Universal JMS sessions, which would allow queues and topics to be updated within the same transaction.
- Storing temporary log records in a topic is not practical.

Outbound

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

Connection Types

MQ Series supports two connection types:

- Local (bind)
- Remote (client)

Local Connections

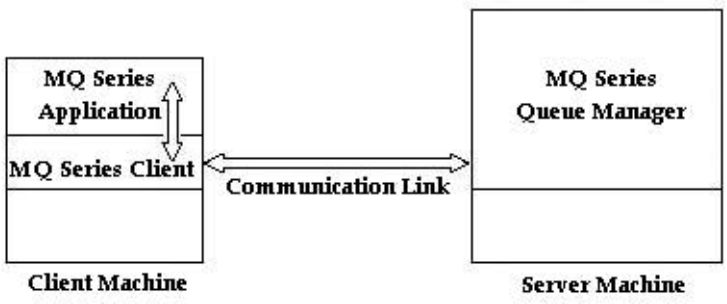
Local connections imply that the MQ Series queue manager is running on the same host as the MQ Series adapter. In this case, the MQ Series adapter only needs to know the name of the queue manager and queue name itself in order to establish a queue connection.

Remote Connections

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

Figure 1–2 displays a client connection:

Figure 1–2 Client Connection



MQ Series Adapter Limitations

The MQ Series adapter has the following limitations:

- All messages received are assumed to be unencrypted, as the MQ Bridge is not expected to do any decryption before handing messages to the D3L engine.
- All but the ObjectMessage JMS message types are supported: TextMessage, BytesMessage, MapMessage and StreamMessage.
- All MQ Series Report messages are used only to the extent MQ Series expects it but otherwise they are not utilized.
- MQ Series message grouping and segmentation are not supported.
- JMS Message properties of received messages from MQ Series are not passed on to Oracle9*i*AS InterConnect. To some extent they can be used in selecting a relevant D3L transformation.
- MQ Series transactions are used to support the Oracle9*i*AS InterConnect TransactionalMessageReceiver interface. The `mq.default.trans_id_expiry` configuration parameter determines how long a transaction started by the Oracle9*i*AS InterConnect Agent, can stay idle before it expires. MQ Series does not expose the concept of a persistent transaction identifier, therefore, the transaction identifier is only valid for the lifespan of the MQ Series adapter instance and the underlying transactional JMS session. Consequently, a given transaction ID is rendered invalid immediately when the adapter process dies.

- An MQ Series adapter instance only supports one outgoing (sending) endpoint. For example, it can only communicate with one queue manager.

2

Installation and Configuration

This chapter discusses the following topics:

- Installing the MQ Series Adapter

Installing the MQ Series Adapter

This section contains these topics:

Preinstallation Tasks

The MQ Series 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)

Before installation, make sure that the MQ Series server and possibly the MQ Series client if the server is running on a remote host, are installed.

For IBM installation guides, see the following:

- MQSeries for Sun Solaris, V5R2 Quick Beginnings—`http://www-4.ibm.com/software/ts/mqseries/library/manualsa/amqdac03/amqdac03tfrm.htm`
- MQSeries for Windows NT and Windows 2000 V5R2 Quick Beginnings—`http://www-4.ibm.com/software/ts/mqseries/library/manualsa/amqtac01/amqtac01tfrm.htm`
- MQSeries for HP-UX V5.2 Quick Beginnings—`http://www-3.ibm.com/software/ts/mqseries/library/manualsa/amqcac03tfrm.htm`
- README for MQSeries for HP-UX V5.2—`http://www-4.ibm.com/software/ts/mqseries/support/readme/hpx520_read.html`
- MQSeries for AIX V5.2 Quick Beginnings GC33-1867-03—`http://www-3.ibm.com/software/ts/mqseries/library/manualsa/amqaac03/amqaac03tfrm.htm`
- README for MQSeries for AIX V5.2—`http://www-4.ibm.com/software/ts/mqseries/support/readme/aix520_read.html`

- MQSeries for Compaq Tru64 UNIX, V5.1 Quick Beginnings—`http://www-4.ibm.com/software/ts/mqseries/library/manualsa/amq2ac/amq2ac.htm`
- IBM MQSeries for Tru64 UNIX, Version 5.1 READ.ME—`http://www-4.ibm.com/software/ts/mqseries/support/readme/tru64_read.html`
- MQSeries for Linux V5.2 Quick Beginnings GC34-5691-01—`http://www-3.ibm.com/software/ts/mqseries/library/manualsa/amq1ac01/amq1ac01tfrm.htm`
- README for MQSeries for Linux V5.2—`http://www-4.ibm.com/software/ts/mqseries/support/readme/linux520_read.html`

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

Verify that the MQ Series installation is working before commencing the installation of the MQ Series adapter.

See Also: Verifying the Installation in all IBM guides.

Consult the Oracle9i Application Server Installation Guide before proceeding with MQ Series adapter installation. This guide includes information on:

- CD-ROM mounting
- Oracle Universal Installer startup
- Oracle9*i*AS InterConnect software, hardware, and system requirements
- Oracle9*i*AS InterConnect installation

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 MQ Series adapter installation.

Installation Tasks

To install the MQ Series adapter:

1. Click Next on the Welcome page.

   The File Locations page displays.

2. Enter the following information in the Destination fields:

   - Name—The Oracle home name.
   - Path—The full path to the Oracle home in which to install the MQ Series adapter.

   Note: Do not change the path specified in the Source field. This is the location on the CD-ROM from which to install the MQ Series adapter.

3. Click Next.

   The Installation Types page displays.

4. Select Oracle9*i*AS InterConnect Adapters and click Next.

   The Available Product Components page displays.

5. Select Oracle9*i*AS InterConnect MQ Series Adapter and click Next.

6. If the MQ Series adapter is not being installed on the same computer as Oracle9*i*AS InterConnect and another adapter is not installed in the current Oracle home, the Oracle9*i*AS InterConnect Hub Database page displays. Enter the following information about the Oracle9*i*AS InterConnect Hub to use:

   - Host name—The hostname of the computer on which Oracle9*i*AS InterConnect is installed.
   - Port Number—The port number of the computer.
   - Database SID—The system identifier (SID) of the Oracle9*i*AS InterConnect Oracle9*i*AS Metadata Repository.
   - Password—The password for the Oracle9*i*AS Metadata Repository schema.

   The Oracle9*i*AS Metadata Repository stores metadata used by Oracle9*i*AS InterConnect to coordinate communication between components.

7. Click Next.

   The Oracle9*i*AS InterConnect MQ Series Adapter Configuration page displays.

8. Enter the name of the application associated with the MQ Series adapter. White spaces or blanks are not allowed. This is the same application name created or to be created in iStudio. The default value is `myMQseriesApp`.

9. Click Next.

   The Oracle9*i*AS InterConnect MQ Series Adapter usage page displays.

10. Select one of the following options and go to the step specified to enable the sending and/or receiving of messages from an external data source, such as an MQ Series server. You can change your selections later by editing parameter settings in the `adapter.ini` file.

If You Select...	Then Click Next and Go to Step...
Configure for both sending and receiving messages	11
Configure for sending messages ONLY	11
Configure for receiving messages ONLY	13

11. Enter the following information in the Oracle9*i*AS InterConnect MQ Series Adapter Configuration - Configure sending endpoint information page:

    - MQ Series inbound queue—The URI of the MQ Series queue to which messages are sent.
    - MQ Series inbound log queue—The URI of the MQ Series queue which temporarily stores log records during sending transactions. This queue and the next can refer to the same physical MQ Series queue.
    - MQ Series inbound id queue—The URI of the MQ Series queue which is used to store and generate unique (sequential) transaction identifiers for the inbound or sending transactions.

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

12. Click Next.

    The installation page that displays next is based on the selection you made in Step 10:

If You Selected...	Then Go to Step...
Configure for both sending and receiving messages	13
Configure for sending messages ONLY	14

13. Enter the MQ Series outbound queue and topic information. This is a URI for the MQ Series queue or topic from which messages are receiving. It is used for listening to incoming messages from MQ Series or as JMS ReplyTo addresses when sending request messages to MQ Series.

14. Enter the following information on the Oracle9*i*AS InterConnect MQ Series Adapter – Define MQ Series connection information page:

    - MQ Series Java installation path—This path specifies the root directory of the MQ Series Java (client) installation, which typically is `/opt/mqm/java`. At the startup time, the MQ Series Adapter will look in the `lib` subdirectory of this path to find the JAR files `jms.jar`, `com.ibm.mqjms.jar`, and `com.ibm.mq.jar`.
    - MQ Series Queue Manager—The name of the MQ Series queue manager to which to connect.
    - MQ Series Client Connection Type—From the drop down list, select the type of connection to make to the MQ Series queue manager. Select Remote to use a client connection (through an MQ Series channel), or select Local to bind to a queue manager running on the same machine as the adapter.

15. Click Next.

    The installation page that displays next is based on the selection you made in Step 14:

If You Selected...	Then Go to Step...
Remote	16
Local	17

16. Enter the following information on the Oracle9*i*AS InterConnect MQ Series Adapter – Specify MQ Series client connection parameters page:
    - Host Name—The DNS name of the host where the MQ Series queue manger resides.
    - Port Number—The port number to connect to on the MQ Server host. The default port number is 1414. This port is defined when starting the MQ Series listener via the command `runmqlsr` (for example, `runmqlsr -m <qmqr> -t tcp -p 1415`).
    - MQ Series Channel Name—The name of the MQ Series channel to use for the client connection.
17. Click Next.
18. Complete any other fields for other components selected for installation, such as other adapters.

    When finished, the Summary page displays.
19. Click Install to install the MQ Series adapter. The 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`*

`Application` is the value you specified in Step 8 on page 2-5.

Configuring the MQ Series 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 MQ Series 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)	
`ignoreErrors.bat` (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 `ErrorCodes.ini` will be ignored.
`ignoreErrors` (UNIX)	

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.

File	Description
`d3l-file.xml`	One or more D3L XML files that describe the mappings between MQ Series native / binary fixed-structure messages and Oracle9*i*AS InterConnect Application View messages.

Table 2–4 Directories

File	Description
persistence	The messages are persisted in this directory. This directory or its contents should not be 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 MQ Series 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 Files

Hub.ini

The MQ Series 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). The possible value is a valid hub database username. The default value is NONE.	`hub_username=myhub`
`hub_password`	The password for the hub database user. The possible value is a the valid password for the hub database user. The default value is NONE.	`hub_password=manager`
`hub_host`	The name of the machine hosting the hub database. The possible value is a the valid machine name. The default is NONE.	`hub_host=mpjoshipc`
`hub_instance`	The valid SID of the hub database. The possible value is a valid SID. The default is NONE.	`hub_instance=orcl`
`hub_port`	The TNS listener port number for the HUB database instance. The possible value is a TNS listener port number. The default value is NONE.	`hub_port=1521`
`repository_name`	The name of the repository this adapter talks to. The possible value is a valid repository name. The default value is NONE.	`repository_name=myrepo`

Adapter.ini

The MQ Series 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 to which this adapter connects. 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=mqapp`
`partition`	The partition this adapter handles as specified in iStudio. Any alphanumeric string can be used. 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: 0=errors only 1=status and errors 2=trace, status, and errors 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 Queueing subscriber name. There is no default value.	`agent_subscriber_ name=mqapp`
`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 '%mqapp,%'`
`agent_reply_ subscriber_name`	The subscriber name is 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=mqapp, instance_number=2, then, agent_reply_ subscriber_name=mqapp2`

Parameter	Description	Example
agent_reply_ message_ selector	Used only if there are 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=mqapp, instance_number=2, then agent_reply_ message_ selector=receipient_ list like'%,mqapp2,%'
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 and 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 and 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 and false. There is no default value.	agent_use_custom_hub_ dtd=false
agent_metadata_ caching	Specifies the metadata caching algorithm. Possible values are: ■ 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

Parameter	Description	Example
agent_dvm_table_caching	Specifies the DVM caching algorithm. Possible values are: ■ 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. The default value is demand.	agent_dvm_table_caching=demand
agent_lookup_table_caching	Specifies the lookup table caching algorithm. Possible values are: ■ 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. There 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. Set this parameter to true to delete all cached metadata on startup. 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. Possible values are true and false. There 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 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 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. There default value is 60000.	agent_persistence_retry_interval=60000
service_path	Windows only. The value that the environment variable PATH should be 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. Possible values are 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 service_vm_arg2=oai.adapter=.mq

MQ Series Adapter Parameters

The following table lists the parameters specific to the MQ Series adapter.

Parameter	Description	Example
bridge_class	Indicates the entry class for the MQ Series adapter. Do not modify this value. The only possible value is oracle.oai.agent.adapter.mqseries.MQBridge. There is no default value.	bridge_class=oracle.oai.agent.adapter.mq.MQBridge
ota.type	This defines the message type that the Oracle9*i*AS InterConnect Technology Adapter will handle for both incoming and outgoing messages. Possible values are XML and D3L. The default value is XML.	ota.type=D3L

Parameter	Description	Example
mq.default.sender.destination.uri	A URI for the MQ Series inbound queue to which messages will be sent from Oracle9*i*AS InterConnect. The syntax for the URI is described in the MQ Series Using Java guide. Possible values are any valid JMS queue URIs. There is no default value.	mq.default.sender.destination.uri=queue:///INBOUND.QUEUE?priority=1
mq.default.sender.log_queue.uri	URI for the MQ Series log queue used during send transactions. Possible values are any valid JMS queue URIs. There is no default value.	mq.default.sender.log_queue.uri=queue:///OAI.LOG.QUEUE
mq.default.sender.seq_queue.uri	URI for the MQ Series transaction id (sequence generator) queue used during send transactions. It can refer to the same queue as mq.default.sender.log_queue.uri. Possible values are any valid JMS queue URIs. There is no default value.	mq.default.sender.log_queue.uri=queue:///OIA.SEQ.QUEUE
mq.default.receiver.destination.uri	A URI for the MQ Series outbound queue or topic from which messages will be received. Used for listening to incoming messages or as a ReplyTo address when sending request messages to MQ Series. Possible values are any valid JMS queue URIs. There is no default value.	mq.default.receiver.destination.uri=topic://SAP/Events/HR/newEmployee
mq.default.receiver.selector	JMS selector expression applied while dequeueing from the receiver destination. Possible values are any valid JMS selector expressions. There is no default value.	mq.default.receiver.selector=JMS_IBM_Format <> 'MQSTR' AND JMSXUserID = 'scott'
mq.default.receiver.transacted	Specifies whether or not the JMS sessions for the receive URI should be transacted. The JMS session for the sender URI is always transacted. The possible value are Y and N. The default value is N.	mq.default.receiver.transacted=Y
mq.default.receiver.durable	If the receiver.destination.uri parameter specifies a JMS topic, then this parameter defines whether or not a durable subscriber should be used to subscribe to the topic. The possible values are Y and N. The default value is N.	mq.default.receiver.durable=Y
mq.default.receiver.exception.uri	A URI for a MQ Series queue to which unparsable outbound (native) messages will be enqueued. Possible values are any valid JMS queue URIs. The default value is blank.	mq.default.receiver.exception.uri=queue:///EXCEPTION.QUEUE
mq.default.queue_manager	The name of the MQ Series queue manager to connect to. The possible value is any MQ Series queue manager name. There is no default value.	mq.default.queue_manager=mars.queue.manager

Parameter	Description	Example
mq.default. connection_type	The type of connection to make to an MQ Series queue manager. The possible values are bind (local) or client (remote). There is no default value.	mq.default.connection_ type=client
mq.default.channel	The name of the MQ Series channel to use for the client connection. The possible value is any valid MQ Series channel name. The default value is blank.	mq.default.channel=SYSTEM .DEF.SVRCONN
mq.default.hostname	The DNS name of the host where the queue manager resides. The possible value is a valid hostname that can be reached over the network from the MQ Series adapter. The default value is blank.	mq.default.hostname= mqsvrhost1.acme.com
mq.default.port	The port to connect to on the MQ Server host (IBM's default is 1414). The possible value is a valid port number for the MQ Series listener. The default value is blank.	mq.default.port=1414
mq.default.user	MQ Series user ID when connecting to the queue manager. Equivalent to the MQ Series environment variable MQ_USER_ID. The value may be used to verify the identity of the MQ Series adapter. The possible value is a valid MQ Series username. The default value is blank.	mq.default.user=mqm
Encrypted_ mq.default. password	MQ Series (encrypted) password when connecting to the queue manager. Equivalent to the MQ Series environment variable MQ_ PASSWORD. The value may be used to verify the identity of the MQ Series adapter.	Encrypted_ mq.default.password=11241 10710711065108010941084107 731070107110811069
mq.default.receive_ exit	Fully qualified class name of the receive exit being used. The possible value is the classname of a Java class that implements com.ibm.mq.MQReveiveExit. The default value is blank.	mq.default.receive_ exit=mypackage. myReceiveExit
mq.default.send_exit	Fully qualified class name of the send exit being used. The possible value is the classname of a Java class that implements com.ibm.mq.MQSendExit. The default value is blank.	mq.default.send_ exit=mypackage.mySendExit

Parameter	Description	Example
mq.default.security_exit	Fully qualified class name of the security exit being used. The possible value is the classname of a Java class that implements com.ibm.mq.MQSecurityExit. The default value is blank.	mq.default.security_exit=mypackage.MySecurityExit
mq.default.ccsid	The coded-character-set-ID in use on connections instead of the default. For possible values, see table 16 in the MQ Series Using Java Guide. The default value is blank (~819).	mq.default.ccsid=1208
mq.default.polling_interval	The number of milliseconds between attempts to receive a message. Possible values are 0-java.lang.Long.MAX_VALUE. The default value is 5000.	mq.default.polling_interval=5000
mq.default.trans_id_expiry	The number of milliseconds before an idle transaction identifier will expire. Possible values are 0-java.lang.Long.MAX_VALUE. The default value is 60000.	mq.default.trans_id_expiry=360000
mq.default.event.name	Should be used if the bridge will only handle one single fixed event name for outbound messages (from MQ Series) and none of the other options below are feasible to use. This parameter requires only one D3L file defined below, with an event name exactly matching this hardcoded event name. The possible value is a valid Oracle9*i*AS InterConnect event name. The default value is blank.	mq.default.event.name=Price.update
mq.default.event.use_mq_fmt	If this parameter value is Y, the bridge uses the IBM MQ Series Message Format field as the name of the Oracle9*i*AS InterConnect event. This message field or property is often referred to as: ■ (C)—MQMD Format field (MQFMT) ■ (Java)—com.ibm.mq.jms.JMSC.FORMAT_PROPERTY The possible values are Y and N. The default value is N.	mq.default.event.use_mq_fmt=Y
mq.default.event.property	If the sending external application is able to specify the event name as a message property value, use this parameter to define the name of the message property that will carry the message event name. The possible value is a valid JMS message property name. The default value is blank.	mq.default.event.property=MyApp_OAIEventProperty

Parameter	Description	Example
`mq.default.event.exit`	Allows a custom Java class to be defined to determine which event name the native MQ Series message corresponds to. It is invoked by the bridge, which provides the received JMS message as input, expecting the event name in return (as a String). This Java class must implement the `oracle.oai.agent.adapter.mqseries.MQEventExit` interface. The possible value is the Java class name of a class that implements the `oracle.oai.agent.adapter.mqseries.MQEventExit` interface. The default value is blank.	`mq.default.event.exit= mypackage.myMqEventExit`

3

Design Time and Runtime Concepts

This chapter describes the design time and runtime concepts for the MQ Series adapter.

MQ Series Adapter Design Time Concepts

The MQ Series adapter can handle XML and D3L structured payload, for example:

- Pure XML data—String beginning with `<?xml....`
- Fixed lay-out—Typically binary data described by a D3L XML file.

XML Payload

A DTD can be imported using iStudio that governs how the MQ Series adapter parses a received XML document into an Oracle9*i*AS InterConnect Application View event and back again, and how an inbound Application View message is converted into an XML document. Use the XML message type when defining a new integration point in the event wizards.

The `ota.type` parameter in the `adapter.ini` file must be set to `XML` and not `D3L`.

When the MQ Series adapter operates in the XML payload mode, no transformations between the native view and the application view are performed on the messages that are sent or received through the MQ Series adapter, apart from the implied straight ASCII <-> Java object conversion or parsing. Any XSLT transformations take place before sending an XML document to Oracle9*i*AS InterConnect, or after receiving one from Oracle9*i*AS InterConnect.

D3L Payload

In addition to pure XML documents, the MQ Series adapter also handles messages that conform to D3L data type definitions, which describe data translations between application view messages and native format.

An application based on the MQ Series adapter can use the use the D3L message type and import D3L data types. When these are selected in the correct wizard, messages received or sent by the MQ Series adapter must adhere to the fixed byte level lay-out defined in an D3L XML file.

Importing D3L data types can also be used to define common view data types.

See Also: *Oracle9iAS InterConnect User's Guide*

MQ Series Adapter Runtime Concepts

This section describes the runtime concepts for the MQ Series adapter.

How the MQ Series Adapter Works

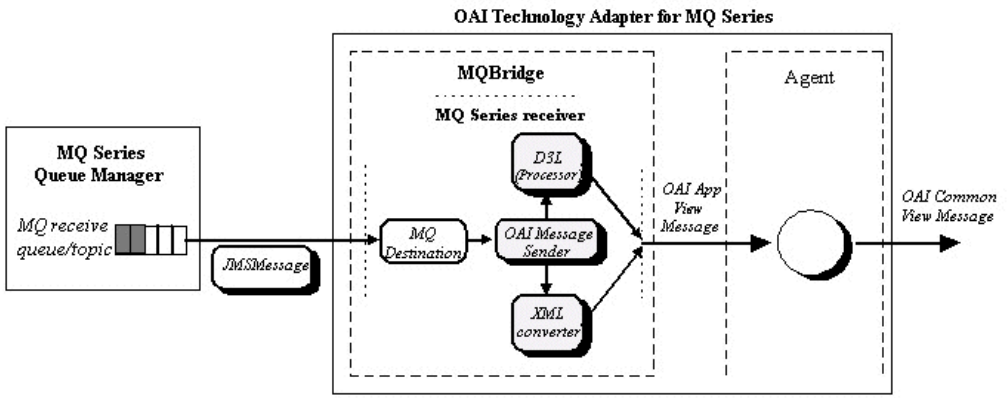
The MQ Series adapter is comprised of the MQ Series bridge and the agent. The following topics describe how the MQ Series adapter works.

Outbound

Outbound messages are processed using the following steps:

1. The bridge is periodically polling the configured MQ Series outbound queue chosen for receiving messages, as defined by the `mq.default.polling_interval` parameter. A new message in this queue indicates a new outbound message from MQ Series waiting to be received by the MQ Series adapter.
2. The adapter then picks up the message, converts it from XML or transforms it via the D3L processor, depending on the value of the `ota.type` parameter, builds the corresponding Oracle9*i*AS InterConnect application view message, and hands it off the Oracle9*i*AS InterConnect agent.
3. The Oracle9*i*AS InterConnect agent then converts the application view event into a common view event and hands it off to the Oracle9*i*AS InterConnect runtime for further routing and processing.

Figure 3–1 Outbound Message Routing



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

See Also: Chapter 2, "Installation and Configuration"

D3L Disambiguation

If the `ota.type` parameter is set to `D3L`, the MQ Series bridge uses the D3L processor to parse from native or byte format to an Oracle9*i*AS InterConnect message object, which then is handed over to the agent as an application view event.

When the MQ Series adapter receives a message from the outbound MQ Series queue while operating in D3L mode, the message is construed as an opaque sequence of bytes. The challenge then becomes how to determine to which Oracle9*i*AS InterConnect event, and ultimately to which D3L file this message corresponds.

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

Note: The term *event name* as used in this section implies a specification of the Oracle9*i*AS InterConnect business object as part of the event name, prefixed followed by a dot, for example, `Order.getStatus`. The event name also synonymously includes Oracle9*i*AS InterConnect procedure names.

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

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

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

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

Using the mq.default.event.property Parameter Use this method if the sending MQ Series application is able to inform the MQ Series adapter about which event a message corresponds to by setting a specified message property to a given value.

To use this method, complete the following:

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

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

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

  - `sap_put.xml`

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

  - `sap_get.xml`

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

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

The `name` and `object` headers should correspond to the associated Oracle9*i*AS InterConnect event and business object names.

Using a D3L Header and Value Pair The MQ Series adapter also supports D3L disambiguation using the header and value attributes. For the MQ Series adapter, transport message headers corresponds to the MQ Series message properties. Consequently, transport message header values means the MQ Series message property values.

See Also: *Oracle9iAS InterConnect User's Guide, Appendix B*

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

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

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

  This requires the following adapter.ini setting:

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

  and the following D3L file:

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

Optionally, if the values in the `MQFMT` field do not easily map into the Oracle9*i*AS InterConnect event names, the user can define a the `mqfmt2event.ini` mapping file in the same directory where `adapter.ini` is located. If present, it will be read and the specified event name mappings applied when a message is received. The format of the file is as follows:

```
<MQMFMT-field-value-1>=<OAI-business-object-name1>.<OAI-event-name1>
<MQMFMT-field-value-2>=<OAI-business-object-name2>.<OAI-event-name2>
...
<MQMFMT-field-value-n>=<OAI-business-object-namen>.<OAI-event-namen>
```

- Example

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

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

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

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

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

- Example: `myEventExit.java`

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

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

- Example: `prod_getprice.xml`

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

If the byte stream of a received message begins with the characters `SYSPR01GETPRC`, the event is resolved as `Product.getPrice` and the shown D3L file is subsequently used to transform the native byte message into an Oracle9*i*AS InterConnect Message Object.

Trying All D3Ls Until One Works If any of the above methods fail, the MQ Series adapter falls back to a trial-and-error resolution scheme where each registered D3L file is tried until one succeeds. This means applying all files in the order they are listed in the `ota.d3ls` parameter in the `adapter.ini` file. If none of the D3L files succeed, the entire D3L disambiguation process for a given message will terminates and an error message is logged. The failed message is saved in the directory where the `adapter.ini` file is located, under a name such as `MQ.FailedMsg.<message-id>`.

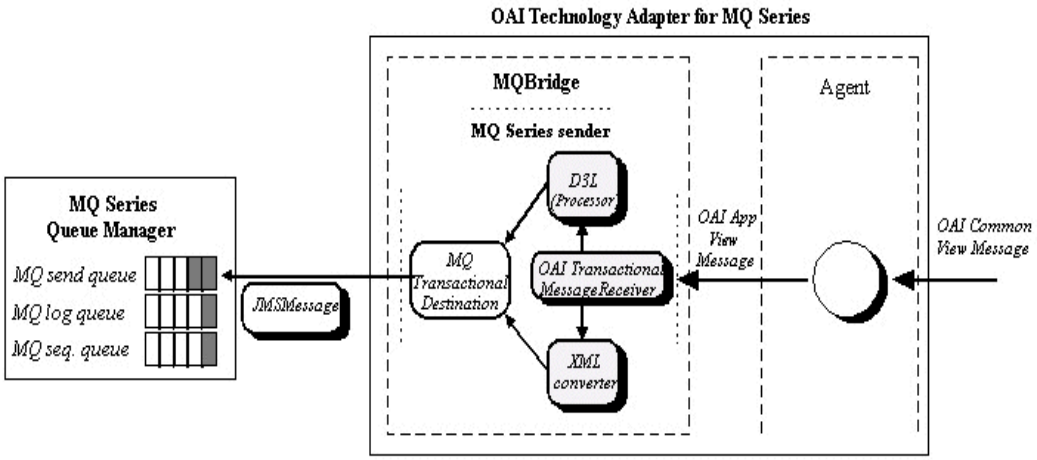
D3L Disambiguation Order The above disambiguation methods are tried in the following order:

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

Inbound

The MQ Series adapter only supports sending to a single MQ Series inbound endpoint, as shown in Figure 3–2. A future edition of the MQ Series adapter will support multiple inbound endpoints.

Figure 3–2 Inbound Message Routing



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

Support for Request-Reply in D3L Mode

The MQ Series adapter supports the Oracle9*i*AS InterConnect PUBLISH, REQUEST, and REPLY message types. Expressed in iStudio terms, the MQ Series adapter can publish or subscribe any event and invoke or implement any procedure.

Note: The HTTP, FTP, and SMTP adapters only support publish and subscribe.

The support for invoke and implement messages, such as Procedure calls, is enabled by the native support for request and reply messages in MQ Series,

including its message correlation capability. It is only available when the MQ Series adapter operates in D3L mode.

In order to take advantage of this capability, a few extra steps need to be performed during configuration, including modifying the D3L files and defining correlation fields in iStudio.

The following instructions are based on a small example:

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

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

getPriceIn.xml

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

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

    <struct id="getPriceInput">

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

</message>
```

getPriceOut.xml

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

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

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

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

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

</message>
```

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

The next paragraphs describe the two possible scenarios, and how to correctly setup Oracle9*i*AS InterConnect to handle them:

- The MQ Series adapter in the role as the invoker.
- The MQ Series adapter in the role as the implementor.

Invoking the Product.getPrice Procedure Using the MQ Series Adapter

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

1. Right-click on Invoked Procedures for the MQ Series application and select New.

   The Invoke Wizard—Select a Procedure page displays.

2. Select getPrice as the Application.
3. Set the Message Type to D3L.
4. Click Next.

   The Define Application View page displays.

5. Click Import and select D3L.

6. Select the `getPriceIn.xml` file and mark as it as IN.
7. Select the `getPriceOut.xml` file and mark as it as OUT.
8. Click OK, then click Finish.
9. Go to the following directory and copy the two XML files (get*.xml) to this directory.

On...	Go to...
Windows	`%ORACLE_HOME%\oai\9.0.2\adapters\<mqapp>`
UNIX	`$ORACLE_HOME/oai/9.0.2/adapters/<mqapp>`

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

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

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

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

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

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

In (native) Invoking Application (JMS example)

```
 // This 3rd party application will send a REQUEST message to
// OAI (Invoke role), and then await a REPLY.
    BytesMessage reqMessage = session.createBytesMessage();
    byte[] getPriceMsg = new byte[] { 20, 0, 0, 0, 10, 0, 0, 0 };
    reqMessage.writeBytes(nativeBytes, 0, nativeBytes.length);
    reqMessage.setJMSReplyTo((Destination)replyQueue);
    reqMessage.setStringProperty("D3Lselector", "getprice");
    reqMessage.setIntProperty("JMS_IBM_MsgType", (int)1); //
REQUEST
```

```
    // Send REQUEST
    queueSender.send(reqMessage);
    session.commit();
    ...
    // Await REPLY
    Message replyMessage = queueReceiver.receive();
    if (replyMessage instanceof BytesMessage)
    {
        if (replyMessage.getJMSCorrelationID().
            equals(reqMessage.getJMSMessageID()))
            // Got my reply back!
```

In (PL/SQL) Implementing Application

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

Which gets invoked from the stub generated by iStudio:

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

Implementing Product.getPrice Procedure Using the MQ Series Adapter

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

1. Right-click on Implemented Procedures for the MQ Series application and select New.

   The Implement Wizard—Select a Procedure page displays.

2. Select getPrice as the Application.

3. Set the Message Type to D3L and click Next.

   The Define Application View page displays.

4. Click Import and select D3L.
5. Select the `getPriceIn.xml` file and mark as it as IN.
6. Select the `getPriceOut.xml` file and mark as it as OUT.
7. Click OK.

   The Define Correlation Fields page displays.

8. Select the two fields in the Input and Output data structures. These fields are used to correlate a response to its original request.
9. Click OK, then click Finish.
10. Go to the following directory and copy the two XML files (get*.xml) to this directory.

On...	Go to...
Windows	`%ORACLE_HOME%\oai\9.0.2\adapters\<mqapp>`
UNIX	`$ORACLE_HOME/oai/9.0.2/adapters/<mqapp>`

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

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

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

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

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

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

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

```
// This 3rd party application will consume/read a REQUEST message from
// OAI (Implement role), and return a REPLY.

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

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

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

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

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


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

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

In (PL/SQL) Invoking Application (Asynchronously)

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

When Oracle9*i*AS InterConnect receives back a reply back from the MQ Series application, it invokes a procedure, for example:

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

Starting the MQ Series Adapter

Start the MQ Series adapter using the `start` script in the directory named after the MQ Series 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.

Sample Log File of Successfully Started MQ Series Adapter

The following file displays an MQ Series adapter that was started successfully:

```
D:\oracle\ora902\oai\9.0.2\adapters\mqapp>D:\oracle\ora902\oai\9.0.2•in\Java
Service.exe -debug "Oracle OAI Adapter 9.0.2 -mqapp"
D:\oracle\ora9021\oai\9.0.2\adapters\mqapp\adapter.ini
The Adapter service is starting..
Registering your application (MQAPP)..
Initializing the Bridge oracle.oai.agent.adapter.mqseries.MQBridge..
Starting the Bridge oracle.oai.agent.adapter.mqseries.MQBridge..
Service started successfully.
```

Stopping the MQ Series Adapter

Stop the MQ Series adapter using the `stop` script in the directory named after the MQ Series 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. 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 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.

4

Frequently Asked Questions

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

- Installation and General Questions
- Design Time Questions

Installation and General Questions

The following questions address installation and other concepts of the MQ Series adapter.

How do I know the MQ Series adapter has started properly?

View the `oai.txt` file located in the appropriate timestamped subdirectory of the MQ Series 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`*

The MQ Series adapter did not start properly - what went wrong?

Inspect the exceptions in the `oailog.txt` file. The exceptions provide information about what went wrong. One reason is that the MQ Series adapter cannot connect to the repository. Make sure the repository is started properly and the MQ Series adapter connects to the repository once it is started properly. You do not need to restart the MQ Series adapter.

> **See Also:** *Oracle9iAS InterConnect User's Guide* for instructions on starting the repository on UNIX and Windows

Can I install multiple MQ Series adapters on the same machine?

The installer overwrites previous installations of the MQ Series adapter if it is installed a second time in the same Oracle home. However, multiple Oracle homes can exist on one machine and have one MQ Series adapter installed in each Oracle home. An Oracle home can be created by installing the Oracle8*i* client in different locations. When the MQ Series adapter is installed a second time, choose an Oracle home different from where the first MQ Series adapter is installed.

Design Time Questions

The following are design time questions for the MQ Series adapter.

When I change an element in iStudio, such as mappings, it seems like the MQ Series adapter is using old information - what is happening?

The MQ Series adapter caches the information from iStudio that is stored in the repository locally for better performance. If you change something in iStudio and want to see the change in the runtime, you need to stop the MQ Series adapter, delete the MQ Series adapter cache files, and restart the MQ Series adapter.

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

How do I secure any of the inifile parameters?

In order to encrypt any values specified in an `.ini` file, complete the following steps:

1. Locate the value to be encrypted.
2. Run the `encrypt` utility in `$OAI_HOME/bin` directory to encrypt the above value.
3. Prefix the name of the parameter in the `.ini` file with `encrypted_`.
4. Replace the value with the new encrypted value from step 2.

Example: To encrypt the password for the `mq.default.password` parameter, replace `mq.default.password=mqm` with the following:

```
encrypted_mq.default.password=1124110710711065108010941084107310701071108110690
```

I am getting a JMS-nnnn error when the MQ Series adapter is starting up - what is wrong?

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

- `2009 MQRC_CONNECTION_BROKEN`—The connection to the queue manager has been lost. This can occur because the queue manager has ended. All previous handles are now invalid, so the MQ Series adapter should be restarted.

- 2030 MQRC_MSG_TOO_BIG_FOR_Q—The message length is greater than the maximum for the queue. Increase MaxMsgLength for the queue (MQ Series Administrator).
- 2031 MQRC_MSG_TOO_BIG_FOR_Q_MGR—The message length is greater than the maximum allowed by the remote queue manager. This reason also occurs if a channel, through which the message is to pass, has restricted the maximum message length to a value that is actually less than that supported by the queue manager and the message length is greater than this value.
- 2035 MQRC_NOT_AUTHORIZED—The user is not authorized to perform the operation attempted. Make sure the mq.default.user and mq.default.password parameters in adapter.ini are correct.

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

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

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

Why am I getting the following error in the log file?

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

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

Why do I get the "Unable to load message catalog - mqji" error message when starting the MQ Series adapter?

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

Index

R

S

T

X