Oracle® Application Server Integration InterConnect

Adapter for DB Installation and User's Guide

10*g* Release 2 (10.1.2)

Part No. B14076-01

November 2004

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

Part No. B14076-01

Primary Author: Vimmy K Raj, Pradeep Vasudev

Contributing Author: Sandeep Jain, Maneesh Joshi, Rahul Pathak, Harish Sriramulu

Contents

Send Us Your Comments v

Preface vii

Audience vii
Documentation Accessibility vii
Structure viii
Related Documents viii
Conventions viii

1 Introduction

1.1 Database Adapter Overview 1-1
1.2 Database Adapter System Requirements 1-1
1.2.1 Hardware Requirements 1-1
1.2.2 Software Requirements 1-1

2 Installation and Configuration

2.1 Installing the Database Adapter 2-1
2.1.1 Preinstallation Tasks 2-1
2.1.2 Installation Tasks 2-1
2.1.3 Verification Test 2-3
2.2 Configuring the Database Adapter 2-3
2.2.1 Using the Application Parameter 2-4
2.2.2 Ini File Settings 2-4
2.2.2.1 hub.ini Parameters 2-4
2.2.2.2 adapter.ini Parameters 2-5

3 Design Time and Runtime Concepts

3.1 Database Adapter Design Time Concepts 3-1
3.1.1 Importing Database Tables and Objects 3-1
3.1.2 Importing Oracle Objects and Advanced Queuing Payloads 3-4
3.1.3 Returned In Arguments 3-4
3.1.4 Deploying PL/SQL Code 3-4
3.2 Database Adapter Runtime Concepts 3-5
3.2.1 How the Database Adapter Works 3-5

3.2.1.1 Database Sender 3-5
3.2.1.2 Database Receiver 3-5
3.3 Starting the Database Adapter 3-5
3.3.1 Log File of Database adapter 3-6
3.4 Stopping the Database Adapter 3-6

4 Sample Use Cases

4.1 Case One: Publish and Subscribe 4-1
4.1.1 Design Time Steps 4-1
4.1.2 Runtime Steps 4-3
4.1.3 Related Files 4-3
4.2 Case Two: Invoke and Implement 4-4
4.2.1 Synchronous Invoke and Implement 4-4
4.2.1.1 Design Time Steps 4-4
4.2.1.2 Runtime Steps 4-6
4.2.2 Related Files for Synchronous Invoke Implement 4-7
4.2.3 Asynchronous Invoke and Implement 4-8
4.2.3.1 Design Time Steps 4-8
4.2.3.2 Runtime Steps 4-10
4.2.4 Related Files for Asynchronous Invoke and Implement 4-10

Index

Send Us Your Comments

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

Part No. B14076-01

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

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

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

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

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

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

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

Preface

This Preface contains these topics:

- Audience
- Documentation Accessibility
- Structure
- Related Documents
- Conventions

Audience

Oracle Application Server Integration InterConnect Adapter for DB Installation and User's Guide is intended for those who perform the following tasks:

- install applications
- maintain applications

To use this document, you need to understand how to install and configure OracleAS Integration InterConnect.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

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

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

This document contains:

Chapter 1, "Introduction"

This chapter describes the OracleAS Integration InterConnect Adapter for DB (Database adapter) and the hardware and software requirements.

Chapter 2, "Installation and Configuration"

This chapter describes installation and configuration of the Database adapter.

Chapter 3, "Design Time and Runtime Concepts"

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

Chapter 4, "Sample Use Cases"

This chapter provides sample use cases for the Database adapter.

Appendix A, "Frequently Asked Questions"

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

Related Documents

For more information, refer to these Oracle resources:

- *Oracle Application Server Integration InterConnect User's Guide*
- *Oracle Application Server Integration InterConnect Installation Guide*

Printed documentation is available for sale in the Oracle Store at

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

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

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

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

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

Conventions

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

- Conventions in Text

- Conventions in Code Examples
- Conventions for Windows Operating Systems

Conventions in Text

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

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

Conventions in Code Examples

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

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

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

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

Convention	Meaning	Example

|	A vertical bar represents a choice of two options.	`{ENABLE | DISABLE}`
		`[COMPRESS | NOCOMPRESS]`

`...`	Ellipsis points mean repetition in syntax descriptions.
	In addition, ellipsis points can mean an omission in code examples or text.
		`CREATE TABLE ... AS` *`subquery`*`;`
		`SELECT` *`col1`*`,` *`col2`*`, ... ,` *`coln`* `FROM employees;`

Other symbols	You must use symbols other than brackets ([ ]), braces ({ }), vertical bars (|), and ellipsis points (...) exactly as shown.	`acctbal NUMBER(11,2);`
		`acct    CONSTANT NUMBER(4) := 3;`

`Italics`	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/`*`system_password`*
		`DB_NAME =` *`database_name`*

`UPPERCASE`	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. Because these terms are not case sensitive, you can use them in either UPPERCASE or lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

`lowercase`	Lowercase typeface indicates user-defined programmatic elements, such as names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Conventions for Windows Operating Systems

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

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

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

1 Introduction

This chapter provides an overview on how to use Oracle Application Server Integration InterConnect (OracleAS Integration InterConnect) Adapter for Databases (Database adapter). It contains the following:

- Database Adapter Overview
- Database Adapter System Requirements

1.1 Database Adapter Overview

The Database adapter enables an Oracle Database application to be integrated with other applications using OracleAS Integration InterConnect. The Database adapter is useful in all Enterprise Application Integration (EAI) scenarios involving Oracle Database applications. The purpose of this guide is to explain all the necessary design time and runtime concepts of the Database adapter. EAI is the integration of applications and business processes within the same company.

1.2 Database Adapter System Requirements

The following sections describe Database adapter system requirements:

- Hardware Requirements
- Software Requirements

1.2.1 Hardware Requirements

Table 1–1 lists the hardware requirements for the computer where the Database adapter will be installed.

Table 1–1 Hardware Requirements

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

1.2.2 Software Requirements

The following sections describe Database adapter software requirements:

- Operating System Requirements
- JRE Requirements

- Database Requirements

Operating System Requirements

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

Table 1–2 Operating System Requirements

Operating System	Version
HP Tru64	HP Tru64 UNIX (Alpha) 5.1b
HP-UX	HP-UX (PA-RISC) 11.11, 11.23
IBM AIX	AIX (POWER) version 5.2
Linux (x86)	Red Hat Enterprise Linux 2.1, 3.0 SuSE SLES8, SLES9
Sun SPARC Solaris	Sun SPARC Solaris 2.8 and 2.9
Microsoft Windows	Windows XP Professional, Windows 2000(SP3 or higher)

JRE Requirements

OracleAS Integration InterConnect uses Java Runtime Environment (JRE) 1.4, which is installed with its components.

Database Requirements

The Database adapter requires Oracle8*i* or later version of the Oracle database. Typically, the database should already be used by the application. If this database is not used by the application, install Oracle8*i*, or Oracle9*i* database.

2 Installation and Configuration

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

- Installing the Database Adapter
- Configuring the Database Adapter

2.1 Installing the Database Adapter

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

This section describes the following topics:

- Preinstallation Tasks
- Installation Tasks
- Verification Test

2.1.1 Preinstallation Tasks

Consult the following guides before installing the Database adapter:

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

2.1.2 Installation Tasks

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

1. In the Available Product Components page of the OracleAS Integration InterConnect installation, select **Database adapter**, and click **Next**.
2. The Set Oracle Wallet Password screen is displayed. Enter and confirm the password on the screen, which will be used to administer OracleAS Integration InterConnect installation. Click **Next**.
    - Go to step 3, if installing the Database adapter in an OracleAS Middle Tier Oracle home that does not have an InterConnect component already installed. Ensure that the OracleAS Integration InterConnect hub has been installed.

- Go to step 4, if installing the Database adapter in an OracleAS Middle Tier Oracle home that has an existing InterConnect component. Ensure that it is a home directory to an OracleAS Integration InterConnect component.

3. The Specify Hub Database Connection screen is displayed. Enter information in the following fields:
   - Host Name: The host name of the computer where the hub database is installed.
   - Port Number: The TNS listener port for the hub database.
   - Database SID: The System Identifier (SID) for the hub database.
   - Password: The password for the hub database user.
4. Click **Next**. The Specify Database Adapter Name page is displayed.
5. Enter the application name. Blank spaces are not permitted. The default value is `myDBApp`.
6. Click **Next**. The Specify Spoke Database Connection Information page is displayed. Enter information in the following fields:
   - Host Name: The name of the computer where the application database is installed.
   - Port Number: The database TNS listener port.
   - Database SID: The SID for the application database.
   - Sys Password: The password of the sys user in the spoke database.

   The information on this page is for the database on the application side from which the adapter will deliver or receive messages. This is not the information for the hub database.
7. Click **Next**. The Spoke Application Database Username page is displayed. Enter information in the following fields:
   - Schema Name: The user name of the user in the Spoke Database.
   - Password: The password for the user name.
8. Click **Next**. The Set Bridge Schema Password page is displayed.
9. Enter and confirm the password for the bridge schema on the screen.
10. Click **Next**. The Summary page is displayed.
11. Click **Install** to install the Database adapter and other selected components. The Database adapter is installed in the following directory:

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

Application is the value you specified in Step 5.

2.1.3 Verification Test

When completing the post-installation steps, no errors should occur. If there are errors, then verify that in the specified database the application using the `oai` schema is the only occurrence. Errors can occur if a Database adapter from previous version installation is talking to this same database.

2.2 Configuring the Database Adapter

After an Database adapter installation, you can configure it for your needs.. The following tables describe the location and details of the configuration files.

Table 2–1 describes the location where the adapter is installed.

Table 2–1 Oracle9i Database Server Adapter Directory

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

Table 2–2 describes the various executable files available for the Database adapter.

Table 2–2 Executable Files

File	Description
`start` (UNIX)	Does not use parameters, starts the adapter.
`start.bat` (Windows)	Does not use parameters, starts the adapter.
`stop` (UNIX)	Does not use parameters, stops the adapter.
`stop.bat` (Windows)	Does not use parameters, stops the adapter.

Table 2–3 describes the Database adapter configuration files.

Table 2–3 Configuration Files

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

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

Table 2–4 Directories

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

2.2.1 Using the Application Parameter

Adapters do not have integration logic. The Database adapter has a generic transformation engine that uses metadata from the repository as runtime instructions to perform transformations. The application parameter defines the capabilities of an adapter, such as the messages to be published and subscribed, and the transformations to be performed. The application parameter allows the adapter to retrieve only the relevant metadata from the repository. The application parameter must match the corresponding application name that will be defined in iStudio under the Applications folder.

If you use prepackaged metadata, then import it into the repository and start iStudio to find the corresponding application under the Applications folder. You can use this as the application name for the adapter you are installing.

2.2.2 Ini File Settings

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

- hub.ini Parameters
- adapter.ini Parameters

2.2.2.1 hub.ini Parameters

The Database adapter connects to the hub database using parameters in the `hub.ini` file located in the `hub` directory. Table 2–5 lists the parameter names, descriptions for each parameter, and examples.

Table 2–5 hub.ini Parameters

Parameters	Description	Example
hub_host	The name of the computer hosting the hub database. There is no default value. The value is set during installation.	`hub_host=mpscottpc`
hub_instance	The SID of the hub database. There is no default value. The value is set during installation.	`hub_instance=orcl`
hub_port	The TNS listener port number for the hub database instance. There is no default value. The value is set during installation.	`hub_port=1521`
hub_username	The name of the hub database schema (or user name). There default value is ichub.	`hub_username=ichub`
repository_name	The name of the repository that communicates with the adapter. The default value is `InterConnectRepository`.	`repository_name=InterConnectRepository`

Oracle Real Application Clusters hub.ini Parameters

When a hub is installed on an Oracle Real Application Clusters database, the parameters listed in Table 2–6 represent information on additional nodes used for connection and configuration. These parameters are in addition to the default parameters for the primary node. In Table 2–6, `x` represents the node number. The number is between 2 and the number of nodes. For example, if the cluster contains 4 nodes, `x` can be a value between 2 and 4.

Table 2–6 Oracle Real Application Clusters hub.ini Parameters

Parameter	Description	Example
hub_hostx	The host where the Real Application Clusters database is installed.	`hub_host2=dscott13`
hub_instancex	The instance on the respective node	`hub_instance2=orcl2`
hub_num_nodes	The number of nodes in a cluster.	`hub_num_nodes=4`
hub_portx	The port where the TNS listener is listening..	`hub_port2=1521`

2.2.2.2 adapter.ini Parameters

The Database adapter connects to the spoke application using parameters in the `adapter.ini` file. Table 2–7 lists the parameter names, descriptions for each parameter, and examples.

Table 2–7 adapter.ini Parameters

Parameter	Description	Example
agent_admin_port	Specifies the port through which the adapter can be accessed through firewalls. Possible Value: A valid port number. Default Value: None.	`agent_admin_port=1059`
agent_delete_file_ cache_at_startup	Specifies whether to delete the cached metadata during startup. If any agent caching method is enabled, then metadata from the repository is cached locally on the file system. Set the parameter to `true` to delete all cached metadata on startup. Possible Values: `true` or `false`. Default Value: `false`. **Note:** After changing metadata or DVM tables for the adapter in iStudio, you must delete the cache to guarantee access to new metadata or table information.	`agent_delete_file_ cache_at_ startup=false`
agent_dvm_table_ caching	Specifies the Domain Value Mapping (DVM) table caching algorithm. Possible values: ■ `startup`: Cache all DVM tables at startup. This may be time-consuming if there are many tables in the repository. ■ `demand`: Cache tables as they are used. ■ `none`: No caching. This slows down performance. Default Value: `demand`.	`agent_dvm_table_ caching=demand`
agent_log_level	Specifies the amount of logging necessary. Possible values: `0`=errors only `1`=status and errors `2`=trace, status, and errors Default Value: 1.	`agent_log_level=2`

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

Parameter	Description	Example
agent_lookup_table_caching	Specifies the lookup table caching algorithm. Possible values: ■ `startup`: Cache all lookup tables at startup. This may be time-consuming if there are many tables in the repository. ■ `demand`: Cache tables as they are used. ■ `none`: No caching. This slows down performance. Default Value: `demand`.	`agent_lookup_table_caching=demand`
agent_max_ao_cache_size	Specifies the maximum number of application object metadata to cache. Possible Value: An integer greater than or equal to `1`. Default Value: `200`.	`agent_max_ao_cache_size=200`
agent_max_co_cache_size	Specifies the maximum number of common object metadata to cache. Possible Value: An integer greater than or equal to `1`. Default Value: `100`.	`agent_max_co_cache_size=100`
agent_max_dvm_table_cache_size	Specifies the maximum number of DVM tables to cache. Possible Value: An integer greater than or equal to `1`. Default Value: `200`.	`agent_max_dvm_table_cache_size=200`
agent_max_lookup_table_cache_size	Specifies the maximum number of lookup tables to cache. Possible Value: Any integer greater than or equal to `1`. Default Value: `200`.	`agent_max_lookup_table_cache_size=200`
agent_max_message_metadata_cache_size	Specifies the maximum number of message metadata (publish/subscribe and invoke/implement) to cache. Possible Value: An integer greater than or equal to `1`. Default Value: `200`.	`agent_max_message_metadata_cache_size=200`
agent_max_queue_size	Specifies the maximum size internal OracleAS Integration InterConnect message queues can grow. Possible Value: An integer greater than or equal to `1`. Default Value: `1000`.	`agent_max_queue_size=1000`
agent_message_selector	Specifies conditions for message selection when the adapter registers its subscription with the hub. Possible Value: A valid Oracle Advanced Queue message selector string (like `'%,aqapp,%'`). Default Value: None.	`agent_message_selector=%,aqapp,%`
agent_metadata_caching	Specifies the metadata caching algorithm. Possible values: ■ `startup`: Cache everything at startup. This may be time-consuming if there are many tables in the repository. ■ `demand`: Cache metadata as it is used. ■ `none`: No caching. This slows down performance. Default Value: `demand`.	`agent_metadata_caching=demand`

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

Parameter	Description	Example
agent_persistence_ cleanup_interval	Specifies how often to run the persistence cleaner thread in milliseconds. Possible Value: An integer greater than or equal to `30000` milliseconds. Default Value: `60000`.	`agent_persistence_ cleanup_ interval=60000`
agent_persistence_ queue_size	Specifies the maximum size of internal OracleAS Integration InterConnect persistence queues. Possible Value: An integer greater than or equal to `1`. Default Value: `1000`.	`agent_persistence_ queue_size=1000`
agent_persistence_ retry_interval	Specifies how often the persistence thread retries when it fails to send an OracleAS Integration InterConnect message. Possible Value: An integer greater than or equal to `5000` milliseconds. Default Value: `60000`.	`agent_persistence_ retry_interval=60000`
agent_pipeline_from_ hub	Specifies whether to turn on the pipeline for messages from the hub to the bridge. If you set the pipeline to `false`, then the file persistence is not used in that direction. Possible Value: `true`, `false` Default Value: `false`.	`agent_pipeline_from_ hub=false`
agent_pipeline_to_ hub	Specifies whether to turn on the pipeline for messages from the bridge to the hub. If you set the pipeline to `false`, then the file persistence is not used in that direction. Possible Value: `true`, `false`. Default Value: `false`.	`agent_pipeline_to_ hub=false`
agent_reply_message_ selector	Specifies the application instance to which the reply must be sent. This parameter is used if multiple adapter instances exist for the given application and given partition. Possible Value: A string built using the application name (parameter:application) concatenated with the instance number (parameter:instance_number). Default Value: None.	If `application=aqapp`, `instance_number=2`, then `agent_reply_message_ selector=`recipient_list `like'%,aqapp2,%'`
agent_reply_ subscriber_name	Specifies the subscriber name used when multiple adapter instances are used for the given application and given partition. This parameter is optional if only one instance is running. Possible Value: The application name (parameter:application) concatenated with the instance number (parameter:instance_number). Default Value: None.	If `application=dbapp` and `instance_number=2`, then `agent_reply_ subscriber_ name=dbapp2`
agent_subscriber_ name	Specifies the subscriber name used when this adapter registers its subscription. Possible Value: A valid Oracle Advanced Queue subscriber name. Default Value: None.	`agent_subscriber_ name=dbapp`

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

Parameter	Description	Example
agent_throughput_ measurement_ enabled	Specifies if the throughput measurement is enabled. Set this parameter to `true` to turn on throughput measurements. Default Value: `true`.	`agent_throughput_ measurement_ enabled=true`
agent_tracking_ enabled	Specifies if message tracking is enabled. Set this parameter to `false` to turn off tracking of messages. Set this parameter to `true` to track messages with tracking fields set in iStudio. Default Value: `true`.	`agent_tracking_ enabled=true`
agent_use_custom_ hub_dtd	Specifies whether to use a custom DTD for the common view message when handing it to the hub. By default, adapters use a specific OracleAS Integration InterConnect DTD for all messages sent to the hub. Set this parameter to `true` to have the adapter use the DTD imported for the message of the common view instead of the OracleAS Integration InterConnect DTD. Default Value: None.	`agent_use_custom_hub_ dtd=false`
application	Specifies the name of the application to which this adapter connects. This must match the name specified in iStudio while creating metadata. Possible Value: An alphanumeric string. Default Value: None.	`application=dbapp`
encoding	Specifies the character encoding for published messages. The adapter uses this parameter to generate encoding information for the encoding tag of transformed OracleAS Integration InterConnect messages. OracleAS Integration InterConnect represents messages internally as XML documents. Possible Value: A valid character encoding. Default Value: `UTF-8`. When there is no existing encoding in the subscribed message, this parameter will be used to explicitly specify the encoding of the published message. This parameter will be ignored when the encoding already exists in the subscribed message.	`encoding=Shift_JIS`
external_dtd_base_url	Specify the base URL for loading external enitites and DTDs.This specifies to the XML parser to resolve the external entities in the instance document using the given URL. Possible Value: A URL. Default Value: The URL of the current user directory.	external_dtd_base_ url=file://C:\InterConnect1 0_1_2\adapters\AQApp\
instance_number	Specifies the instance number to which this adapter corresponds. Specify a value only if you have multiple adapter instances for the given application with the given partition. Possible Value: An integer greater than or equal to `1`. Default Value: None.	`instance_number=1`

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

Parameter	Description	Example
nls_country	Specifies the ISO country code. The codes are defined by ISO-3166. Possible Value: A valid code. A full list of the codes is available at `http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html` Default Value: `US`. **Note**: This parameter specifies date format and is applicable for the date format only.	`nls_country=US`

nls_date_format	Specifies the format for a date field expressed as a string.
	
	Possible Value: A valid date format pattern as shown in Table 2–8 for the definitions of the format characters.
	
	Default Value: `EEE MMM dd HHmmss zzz yyyy`.
		Date format pattern `dd/MMM/yyyy` can represent 01/01/2003.
		
		`nls_date_format=dd-MMM-yy`
		
		Multiple date formats can be specified as `num_nls_formats=2`
		
		`nls_date_format1=dd-MMM-yy`
		
		`nls_date_format2=dd/MMM/yy`

nls_language	Specifies the ISO language code. The codes are defined by ISO-639. Possible Value: A valid code. A full list of these codes is available at `http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt` Default Value: `en`. **Note**: This parameter specifies date format and is applicable for the date format only.	`nls_language=en`
partition	Specifies the partition this adapter handles as specified in iStudio. Possible Value: An alphanumeric string. Default Value: None.	`partition=germany`
service_class	Specifies the entry class for the Windows service. Possible Value: `oracle/oai/agent/service/AgentService`. Default Value: None.	`service_class=oracle/oai/agent/service/AgentService`
service_classpath	Specifies the class path used by the adapter JVM. If a custom adapter is developed and the adapter is to pick up any additional jar files, then add the files to the existing set of jar files. Possible Value: A valid `PATH` setting. Default Value: None. This parameter is for Microsoft Windows only.	`service_classpath=D:\oracle\oraic\integration\interconnect\lib\oai.jar;D:\oracle\oraic\jdbc\classes12.zip`

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

Parameter	Description	Example
service_jdk_dll	Specifies the Dynamic Link Library(DLL) that the adapter JVM should use. Possible Value: A valid `jvm.dll`. Default Value: `jvm.dll`. This parameter is for Microsoft Windows only.	`service_jdk_ dll=jvm.dll`
service_jdk_version	Specifies the JDK version that the adapter JVM should use. Possible Value: A valid JDK version number. Default Value: 1.4 This parameter is for Microsoft Windows only.	`service_jdk_ version=1.4`
service_max_heap_ size	Specifies the maximum heap size for the adapter JVM. Possible Value: A valid JVM heap size. Default Value: `536870912`. This parameter is for Microsoft Windows only.	`service_max_heap_ size=536870912`
service_max_java_ stack_size	Specifies the maximum size the JVM stack can grow. Possible Value: A valid JVM maximum stack size. Default Value: Default value for the JVM. This parameter is for Microsoft Windows only.	`service_max_java_ stack_size=409600`
service_max_native_ stack_size	Specifies the maximum size the JVM native stack can grow. Possible Value: A valid JVM maximum native stack size. Default Value: Default value for the JVM. This parameter is for Microsoft Windows only.	`service_max_native_ size=131072`
service_min_heap_ size	Specifies the minimum heap size for the adapter JVM. Possible Value: A valid JVM heap size. Default Value: `536870912`. This parameter is for Microsoft Windows only.	`service_min_heap_ size=536870912`

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

Parameter	Description	Example
service_num_vm_args	Specifies the number of `service_vm_arg`*`number`* parameters specified in JVM. Possible Value: The number of `service_vm_arg`*`number`* parameters. Default Value: None. This parameter is for Microsoft Windows only.	`service_num_vm_args=1`
service_path	Specifies the environment variable `PATH`. The `PATH` variable is set before starting the Java Virtual Machine (JVM). Typically, list all directories that contain necessary DLLs. Possible Value: The valid `PATH` environment variable setting. Default Value: None. This parameter is for Microsoft Windows only.	`service_path=%JREHOME%\bin;D:\oracle\oraic\bin`

service_vm_arg*number*	Specifies any additional arguments to the JVM. For example, to retrieve line numbers in any stack traces, set `service_vm_arg1=java.compiler=NONE`. If a list of arguments exists, then use multiple parameters as shown in the example, by incrementing the last digit by `1`.
	
	Possible Value: A valid JVM arguments.
	
	Default Value: None.
	
	This parameter is for Microsoft Windows only.
		`service_vm_arg1=java.compiler=NONE`
		
		`service_vm_arg2=oai.adapter=.aq`

Table 2–8 shows the reserved characters used to specify the value of the `nls_date_format` parameter. Use the characters to define date formats.

Table 2–8 Reserved Characters for the Value of the nls_date_format Parameter

Letter	Description	Example
`G`	Era designator	`AD`
`y`	Year	`1996` or `96`
`M`	Month in year	`July` or `Jul` or `07`
`w`	Week in year	`27`
`W`	Week in month	`2`
`D`	Day in year	`189`
`d`	Day in month	`10`
`F`	Day of week in month	`Number 2`
`E`	Day in week	`Tuesday` or `Tue`
`a`	a.m./p.m. marker	`P.M.`
`H`	Hour in day (0-23)	`0`
`k`	Hour in day (1-24)	`24`
`K`	Hour in a.m./p.m. (0-11)	`0`
`h`	Hour in a.m./p.m. (1-12)	`12`
`m`	Minute in hour	`30`

Table 2–8 (Cont.) Reserved Characters for the Value of the nls_date_format Parameter

Letter	Description	Example
`s`	Second in minute	`55`
`S`	Millisecond	`978`

Database Adapter-specific Parameters

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

Table 2–9 Database Adapter-specific Parameters

Parameter	Description	Example
bridge_class	Indicates the entry class for the Database adapter. Do not modify this value. Default Value: `oracle.oai.agent.adapter.db.DBBridge`.	`bridge_class=oracle.oai.agent.adapter.db.DBBridge`
db_bridge_instance	The SID of the database instance. Default Value: None.	`db_bridge_instance=orcl`
db_bridge_num_schemas	The number of alternate schemas that this database adapter will fail over to. Possible Values: An integer greater than 0. Default Value: 1.	`db_bridge_num_schemas=1`
db_bridge_schema#_host	The name of the computer hosting the database instance specified by the `db_bridge_schema#_instance`. Default Value: None.	`db_bridge_schema1_host=ssuravar-sun`
db_bridge_schema#_instance	The SID of the database instance. Possible Value: A valid SID. Default Value: None.	`db_bridge_schema1_instance=oiddb1`
db_bridge_schema#_num_readers	The number of database readers corresponding to the schema number. This is the same as the number of reader threads; each thread has its own database session. Possible Value: An integer greater than 0. Default Value: None.	`db_bridge_schema1_num_readers=1`
db_bridge_schema#_num_writers	The number of database writers corresponding to the schema number. This is same as the number of writer threads; each thread has its own database session. Possible Values: An integer greater than 0. Default Value: None.	`db_bridge_schema1_num_writers=1`

db_bridge_schema#_password	The password for the user specified in the `db_bridge_schema`*`schema#`*`_username`.
	Possible value: The password for the corresponding database user.
	Default Value: None.
		`db_bridge_schema1_password=oai`
		`encrypted_db_bridge_schema1_password=112511011064109110871093`

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

Parameter	Description	Example
db_bridge_schema#_ port	The port where the TNS listener is running for the database instance specified by `db_bridge_schema#_ instance` parameter. Possible Value: A valid TNS listener port number. Default Value: None.	`db_bridge_schema1_ port=1521`
db_bridge_schema#_ username	The user name for the schema number *schema#*. The possible values for the schema number are 1 through *db_bridge_num_schemas*. This value should not be modified. Possible value: A valid database user name Default Value: None.	`db_bridge_schema1_ username=oai`
db_bridge_schema#_ writer_password	The password corresponding to the database user specified in Oracle Wallet by the `db_bridge_ schema#_writer_username` parameter. Possible Values: A valid password. Default Value: None. **Note**: All passwords are stored in Oracle Wallet. Refer to "How do I secure my passwords?" for more details on how to modify and retrieve the password using Oracle Wallet.	`db_bridge_schema1_ writer_ password=welcome`
db_bridge_schema#_ writer_use_oracle_ objects	Specifies whether to use Oracle Objects, available in Oracle8 and later releases. Set this to true unless talking to an Oracle 7.x database. Possible Values: `true` or `false`. Default Value: `false`.	`db_bridge_schema1_ writer_use_oracle_ objects=true`
db_bridge_schema#_ writer_username	The user name to be used by this writer to log on to the database as specified by the `db_bridge_schema#_ instance` parameter. Possible Values: A valid database user. Default Value: None.	`db_bridge_schema1_ writer_ username=mydbapp`
db_bridge_sql_ trace	Used to enable or disable the SQL trace facility for all reader and writer database sessions. Setting this to true results in the SQL query `ALTER SESSION SET SQL_ TRACE = TRUE` being run in the session, thus enabling the SQL trace facility. For more information on the SQL trace facility, including how to format and interpret the output, refer to the Oracle Tuning Guide. Possible Values: `true` or `false`. Default Value: `false`.	`db_bridge_sql_trace= true`
db_bridge_use_thin_ jdbc	Indicates whether to use a thin JDBC driver when talking to the database. Possible Values: `true` or `false`. Default Value: `true`.	`db_bridge_thin_ jdbc=true`

Real Application Clusters adapter.ini Parameters for the DB Adapter When the Database adapter is servicing a Real Application Clusters database as the spoke database, parameters listed in Table 2–10 represent information on connection and configuration.

Table 2–10 Real Application Clusters adapter.ini Parameters

Parameter	Description	Example
db_bridge_num_nodes	Indicates the number of nodes in RAC cluster.	`db_bridge_num_nodes=4`
db_bridge_schema1_ host*x*	Indicates host for the node x.	`db_bridge_schema1_ host2=dsunram13`
db_bridge_schema1_ instance*x*	Indicates instance on node x.	`db_bridge_schema1_ instance2=orcl2`
db_bridge_schema1_ port*x*	Indicates port for node x.	`db_bridge_schema1_ port2=1421`

3

Design Time and Runtime Concepts

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

- Database Adapter Design Time Concepts
- Database Adapter Runtime Concepts
- Starting the Database Adapter
- Stopping the Database Adapter

3.1 Database Adapter Design Time Concepts

During design time, the Database adapter maps relationships between application view and common view. The Database adapter can import the following tables and objects for the application view:

- Relational
- Object
- Oracle Object
- Advanced Queuing payload

This section contains the following topics:

- Importing Database Tables and Objects
- Importing Oracle Objects and Advanced Queuing Payloads
- Returned In Arguments
- Deploying PL/SQL Code

3.1.1 Importing Database Tables and Objects

For a database application, the application and common views resemble the underlying database schema, so iStudio allows the creation of a view by importing tables directly from the database.

The following examples show how importing tables into iStudio modifies their structures:

Example 3–1 Importing Relational Tables

Table 3–1 shows a simple relational database table.

Table 3–1 Customer

Parameter	Value
NAME	VARCHAR2(200)
ID	NUMBER
ADDRESSES	LONG

When imported into iStudio, this table appears as shown in Table 3–2.

Table 3–2 Customer

Parameter	Value
NAME	STRING
ID	INTEGER
ADDRESSES	STRING

When importing from database, iStudio allows any number of columns to be selected.

Example 3–2 Object Table

Table 3–3 shows a simple object table.

Table 3–3 Customer

Parameter	Value
NAME	VARCHAR2(200)
ID	NUMBER
ADDRESSES	ADDRESS_ARRAY

Where ADDRESS_ARRAY is VARRAY of ADDRESS and ADDRESS is an OBJECT TYPE containing the following attributes:

Parameter	Value
CITY	VARCHAR2(200)
STATE	VARCHAR2(200)
ZIP	NUMBER

When imported into iStudio, this table appears as shown in Table 3–4.

Table 3–4 Customer

Parameter	Value
NAME	STRING
ID	INTEGER
ADDRESSES	ARRAY (marked as an ARRAY)

Where ADDRESS_ARRAY contains the following attributes:

Parameter	Value
CITY	STRING
STATE	STRING
ZIP	NUMBER

When dealing with Oracle Object Types, the hierarchical structure is kept intact.

Example 3–3 Foreign Key

For FOREIGN keys, you must import each of the different tables and manually set up the relationship in iStudio by editing the types of attributes.

Relational Tables related by a FOREIGN Key

Table 3–5 Customer

Parameter	Value
NAME	VARCHAR2(200)
ID	NUMBER
ADDRESS	NUMBER (Foreign key)

Table 3–6 Address

Parameter	Value
ID	NUMBER (Primary key)
CITY	VARCHAR2(100)
STATE	VARCHAR2(50)
ZIP	NUMBER

Using iStudio, complete the following to import this structure:

1. Import the Address table.This results in the following:

Parameter	Value
ID	NUMBER
CITY	STRING
STATE	STRING
ZIP	NUMBER

2. Import the Customer table. This results in the following:

Parameter	Value
NAME	STRING
ID	NUMBER
ADDRESS	NUMBER

1. Change the type of Address attribute to `Address`.

3.1.2 Importing Oracle Objects and Advanced Queuing Payloads

Importing an Oracle Object or an Advanced Queuing payload in iStudio is similar to importing database tables. Importing from an Advanced Queuing payload is necessary when working with Advanced Queuing applications.

Note: When importing an Advanced Queuing payload, it may be necessary to log in as the `system` user.

3.1.3 Returned In Arguments

The Returned In Args button appears only in the Invoke wizard. Returned In Arguments is used to propagate `INOUT` attributes contained in the request. If this feature doesn't exist, then you have to ensure that these attributes exist in both the common view and application view of the implementing application and are `INOUT` attributes. It would also be necessary to complete all the mappings to copy these attributes on their way out and back in, when receiving the reply. Returned In Args can also be used to correlate the reply with an asynchronous request.

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

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

If this is to be sent as part of a `CreateCustomer` message and `ID` is to be `INOUT` in both the request and the reply, then it should be an `INOUT` parameter. To do this, complete the following steps:

1. Click **Returned In Args** on the Invoke wizard.
2. Select `ID` in the Please Select In Arguments dialog and the Please Select Out Arguments dialog.

3.1.4 Deploying PL/SQL Code

If the Database adapter is used to connect to an application, then iStudio generates PL/SQL stored procedures. These stored procedures enable an application to interface with OracleAS Integration InterConnect through the Oracle database. This code is generated regardless of the integration point used, which is the event for publish/subscribe or procedure for request/reply, and must be deployed in the application schema to be executed at runtime. To deploy PL/SQL code, use the Deploy PL/SQL context menu in iStudio.

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

3.2 Database Adapter Runtime Concepts

The following section describes the runtime concepts pertinent to the Oracle9*i* Database Server.

3.2.1 How the Database Adapter Works

The following topics describe how the Database adapter works.

3.2.1.1 Database Sender

The Database adapter is comprised of the database bridge and the runtime agent. The bridge is constantly polling the `MESSAGEOBJECTTABLE` table in the `oai` schema, specified by the `db_bridge_schema1_username` parameter. A new row in this table indicates a new outbound OracleAS Integration InterConnect message waiting to be sent by this adapter. The adapter then picks up the message from the interface tables residing in the `oai` schema, builds the corresponding OracleAS Integration InterConnect message, persists it, transforms it to the common view, and routes it to the hub. From the hub, the message gets routed to the corresponding subscriber based on configuration completed in iStudio, which can be content-based or subscription-based.

The application and the database adapter communicate through the interface tables residing in the `oai` schema for outbound messages and through iStudio PL/SQL generated procedures for inbound messages. Thus, if the adapter is down while the application is publishing OracleAS Integration InterConnect messages using the iStudio generated PL/SQL procedures, then the messages are held in the interface tables and will be picked up in a FIFO method by the database adapter once it is up and running. If there are messages in the interface tables that no longer need to be published, then the `DELETE FROM MESSAGEOBJECTTABLE` using SQLPlus can be run in the `oai` schema.

3.2.1.2 Database Receiver

On the subscribing/receiving side, the Database adapter receives the message from the hub, transforms it from common view to application view, and passes it to the bridge, which calls the corresponding PL/SQL procedures, to inform the application about the newly arrived message. If this adapter were an implementing application, then the `OUT` arguments from the PL/SQL procedure invocation are put together and the REPLY in the form of another OracleAS Integration InterConnect message is sent back to the `INVOKER` or `REQUESTER`.

The receiving adapter is responsible for creating any necessary cross-reference entries. In a publish-subscribe scenario, the subscribing adapter creates the cross-reference entry using the returned arguments, for example `OUT`, from the subscribe side procedure.

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

3.3 Starting the Database Adapter

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

- To start the Database adapter on UNIX:

  1. Change to the directory containing the start script.

     ```
     cd ORACLE_HOME/integration/interconnect/adapters/Application
     ```

2. Type **`start`** and press **Enter**.

- To start the Database adapter from Services on Windows:

1. Access the Services window from the Start menu. The Services window is displayed.

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

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

On...	Choose...
Windows 2000	Right-click the service and choose start from the menu that appears.

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

3.3.1 Log File of Database adapter

You can verify the start up status by viewing the `oailog.txt` files. The files are located in the timestamped subdirectory of the `log` directory of the Database adapter. Subdirectory names take the following form:

```
timestamp_in_milliseconds
```

The following is an example of the information about a Database adapter that successfully started.

```
The Adapter service is starting..
Registering your application (DBAPP)..
Initializing the Bridge oracle.oai.agent.adapter.database.DBBridge
Starting the Bridge oracle.oai.agent.adapter.database.DBBridge
Service started successfully.
db_bridge_writer_1 has been started.
db_bridge_reader_1 has been started.
db_bridge_writer_1 has connected to the database successfully.
db_bridge_reader_1 has connected to the database successfully.
```

3.4 Stopping the Database Adapter

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

- To stop the Database adapter on UNIX:

1. Change to the directory containing the stop script.

```
cd ORACLE_HOME/integration/interconnect/adapters/Application
```

2. Type **`stop`** and press **Enter**.

- To stop the Database adapter from Services, on Windows:

  1. Access the Services window from the Start menu. The Services window is displayed.

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

  2. Select the **OracleHomeOracleASInterConnectAdapter-Application** service.
  3. Stop the service. Based on the operating system, the method for stopping it varies.

On...	Choose...
Windows 2000	Right-click the service and choose Stop from the menu that appears.

You can verify the stop status of the Database adapter by viewing the `oailog.txt` files. These files are located in the timestamped subdirectory of the `log` directory of the Database adapter.

4 Sample Use Cases

This chapter describes sample use cases for the Database adapter. For all of the scripts and steps for the use cases provided in this chapter, replace the following strings with the correct values.

- `repo_owner`: The repository owner.
- `version`: The version of the metadata in iStudio. This is usually `V1` unless the metadata versioning features was used in iStudio.

4.1 Case One: Publish and Subscribe

This case illustrates a simple Publish-Subscribe scenario using a Database adapter at each end. In this case, a `Customer` message containing the `ID` attribute and an array of Addresses is published using a PL/SQL procedure. This message is picked up by the publishing adapter, published, and routed to the corresponding subscribing adapter through the hub. The message becomes a new row in a table in the destination schema. These adapters can be located anywhere and can talk to any database. The scripts described here create the publish and subscribe side schemas on the same database. These scripts can be modified to fit any custom scenario.

4.1.1 Design Time Steps

The following section describes metadata creation using iStudio.

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

1. Create a Business Object in iStudio. Enter Customer in the Business Object Name field in the Create Business Object dialog.
2. Create a common data type. In the Create Data Type dialog, complete the following:
   a. Enter Address in the Common Data Type Name field.
   b. Add the following attributes in the Name field:
      - city (STRING)
      - state (STRING)
      - zip (STRING)
3. Create an event in iStudio. In the Create Event dialog, complete the following:
   a. Select Customer for the Business Object.

   b. Enter createCustomer in the Event Name field.
   c. Click **Add** to add the following attributes:
      * id (NUMBER)
      * address (Address) [ARRAY]
4. Create an application in iStudio. Enter demopub in the Application Name field in the Create Application dialog.
5. Create a Published Event using the Publish Wizard in iStudio:
   a. Select demopub from the Application drop-down list and Database from the Message Type drop-down list in the Select an Event page.
   b. Expand the list in the Select an Event page and select createCustomer.
   c. Click **Import** in the Define Application View page to import attributes from the Common View.
   d. Create the following mapping for the newCustomer procedure on the Define Mapping IN Arguments page:
      * `createCustomer [demopub View] -- Object Copy -- createCustomer [Common View]`
   e. Click **Finish**.
6. Create an application in iStudio. Enter demosub in the Application Name field in the Create Application dialog.
7. Create a Subscribed Event using the Subscribe Wizard in iStudio.
   a. Select demosub from the Application list and Database from the Message Type drop-down list in the Select an Event page.
   b. Expand the list in the Select an Event page and select createCustomer.
   c. Click **Import** in the Define Application View page and select Common View to import data types from the Common View.
   d. Create the `createCustomer [Common View] -- Object Copy -- createCustomer [demosub View] mappings` on the Define Mappings page.
   e. Enter the following SQL code on the Define Stored Procedure page:
      * For `sub_createCustomer_`*`repo_owner_version`*:
      * Following the line `dummy:= 0;`, Enter `insert into results values (id, address);`
8. Click **Finish**.
9. Export SQL Code using iStudio. In the Export Application dialog, complete the following:
   a. Select demopub and demosub in the Select the Messages or Types of Message to Export box.
   b. Enter demo in the File Prefix field.

      The following files are created and stored in the *`ORACLE_HOME`*`/integration/interconnect/iStudio` directory:
      * `demo_demopub_Customer.sql`
      * `demo_demopub_CustomerTYPES.sql`

- `demo_demosub_Customer.sql`
- `demo_demosub_CustomerTYPES.sql`

4.1.2 Runtime Steps

The following steps are based on the following files:

- `create_demo_users.sql`
- `create_demo_table.sql`
- `demo_publish.sql`

See Also: "Related Files" on page 4-3

To complete the following steps, run the `create_demo_users.sql` file as the `system` user.

1. Bring up two SQL prompts:
   - Connect as the demopub/manager and run `@demo_demopub_CustomerTYPES, @demo_demopub_Customer, @demo_publish`.
   - Connect as dempsub/manager and run `@demo_demosub_CustomerTYPES, @create_demo_table, @demo_demosub_Customer`.
2. Bring up the demopub and demosub adapters:
   - In a publish SQL prompt, run `exec demo_publish(ANY NUMBER)` in the demopub schema. A new row is created in the Results table in demosub schema every time it receives a message from demopub.

Note: If a Database adapter has already been installed with the application name of demopub, use the `copyAdapter` script in the *`ORACLE_HOME`*`/integration/interconnect/bin` directory to create the `demosub` adapter. Usage: `copyAdapter demopub demosub`. Then, manually enter the user name and password for log in.

4.1.3 Related Files

The following files are related to the runtime steps in CASE ONE.

- File: `create_demo_users.sql`

```
CREATE USER demopub identified by manager;
GRANT connect, resource to demopub;
CREATE USER demosub identified by manager;
GRANT connect, resource to demosub;
```

- File: `create_demo_table.sql`

```
CREATE TABLE results (id NUMBER, address demosub_Address_repo_owner_version_
Arr);
```

- File: `demo_publish.sql`

```
CREATE OR REPLACE PROCEDURE Demo_Publish(id NUMBER)
AS
  moid NUMBER;
  aoid NUMBER;
  addrid NUMBER;
BEGIN
  Customer.crMsg_createCustomer_repo_owner_version(moid, aoid, id);
  addrid := Customer.cr_Address_address('SFO', 'CA', '94040', moid, aoid);
  addrid := Customer.cr_Address_address('Reno', 'NV', '93949', moid, aoid);
  addrid := Customer.cr_Address_address('SJC', 'CA', '95117', moid, aoid);
  Customer.pub_createCustomer_repo_owner_version(moid, 'demopub');
  COMMIT;
END;
/
```

4.2 Case Two: Invoke and Implement

This use case illustrates a simple invoke and implement scenario using a Database adapter at each end. Both synchronous and asynchronous modes of invocation are illustrated. A `Customer` message containing the `ID` attribute, and an array of `Addresses` is sent using a PL/SQL procedure. This message is picked up by the invoking adapter and routed to the corresponding implementing adapter through the hub. On the implementing end, a new row is created in a table in destination schema and a response is sent back indicating that it has received this message. Subsequently on receiving the response, the invoking adapter updates the status for the corresponding customer.

These adapters can be located anywhere and can talk to any database. The scripts provided create the sender and receiver side schemas on the same database. These schemas can be modified to adapt to any custom scenario.

4.2.1 Synchronous Invoke and Implement

Run the `demo_setup.sql` file to create necessary schemas in the database on the application or spoke database. It may be necessary to connect as the `system` user.

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

4.2.1.1 Design Time Steps

1. Create a Business Object in iStudio. In the Create Business Object dialog, enter Customer in the Business Object Name field.
2. Create a common data type.
3. Create a procedure in iStudio. In the Create Procedure dialog, complete the following:
   a. Select Customer for the business object.
   b. Enter newCustomer in the Procedure Name field.
   c. Click **Import** and select **Database** to import attributes.
   d. Log in to the Database as the FOO user.
      * Expand the FOO schema, Tables/Views and select `FOO.CUSTOMERS`.
      * In the right hand side of the dialog, select the ID, ADDRESS, and STATUS columns using the control key.
      * Click **Done** to return to the Publish Wizard.

  * Import arguments as IN arguments in the Publish Wizard. Change the last column (IN/OUT/INOUT) for Status to Out and click **Save**.

4. Create an application in iStudio. Enter demoinv in the Application Name field on the Create Application dialog.
5. Create an invoked procedure using the Invoke Wizard in iStudio:
   a. Select demoinv for the Application and Database as the Message Type in the Select a Procedure page.
   b. Expand the list in the Select a Procedure page and select newCustomer.
   c. Click **Import** and select Common View on the Define Application View page to import attributes from the common view.
   d. Change the `ID` attribute from `IN` to `INOUT`.

      See Also: Appendix A, "Frequently Asked Questions"

   e. Check the box for Synchronous.
   f. Click **Returned In Args** and enter the following:
      * In Argument: ID
      * Out Argument: ID
6. Create the following mapping for the newCustomer procedure on the Define Mapping IN Arguments page:
   - `newCustomer:IN [demoinv View] -- Object Copy -- newCustomer:IN [Common View]`
7. Create the following mapping for the newCustomer procedure on the Define Mapping OUT Arguments page:
   - `newCustomer:OUT.STATUS [Common View] -- Copy Fields -- newCustomer:OUT.STATUS [demoinv View]`
8. In the Define Stored Procedure page, do not edit the SQL code, it is correct.
9. Click **Finish**.
10. Create an application in iStudio. In the Create Application dialog, enter demoimp in the Application Name field.
11. Create an implemented procedure using the Implement Wizard in iStudio:
    a. Select demoimp for the Application and Database as the Message Type.
    b. Expand the list in the Select a Procedure page and select newCustomer.
    c. Click **Import** and select Database in the Define Application View page to import attributes from the database.
    d. Enter the correct information on the Database Login dialog for the BAR schema.
       * Expand BAR, Tables/Views and select `BAR.RESULTS`.
       * In the right hand side of the dialog, select the ID, ADDRESS, and STATUS columns using the control key.
       * Click **Done**.
       * Import arguments as IN arguments. Add an attribute called STATUS [String, OUT].

12. Create the following mapping for the newCustomer procedure in the Define Mapping IN Arguments page:
    - `newCustomer:IN [Common View] -- Object Copy -- newCustomer:IN [demoimp View]`
13. Create the following mapping for the newCustomer procedure in the Define Mapping OUT Arguments page:
    - `newCustomer:OUT [dempimp View] -- Object Copy -- newCustomer:OUT [Common View]`
14. Edit the SQL code in the Define Stored Procedure page as follows:
    - For `imp_newCustomer_`*`repo_owner_version`*, following the line `dummy:= 0;`, enter `insert into results values(i_id, i_address);o_ status := 'SUCCESS';`
15. Click **Finish**.
16. To Export SQL code, right-click **Applications** in iStudio, and select **Export PL/SQL**. Select demoinv and demoimp from the context menu.
17. Enter demo for the File Prefix field.

    The following files are created and stored in the *`ORACLE_ HOME`*`/integration/interconnect/iStudio` directory:
    - `demo_demopub_Customer.sql`
    - `demo_demopub_CustomerTYPES.sql`
    - `demo_demosub_Customer.sql`
    - `demo_demosub_CustomerTYPES.sql`

4.2.1.2 Runtime Steps

The Runtime steps are based on the following files:

- `demo_setup.sql`
- `create_sync_invoke.sql`

Note: Create copies of the Database adapter using the `copyAdapter` script named `demoinv` and `demoimp`. Then, manually input the user name and password for log in.

See Also: Related Files for Synchronous Invoke Implement on page 4-7

Bring up two SQL prompts:

1. At the first SQL prompt, connect as foo/manager.
2. Run the following SQL scripts:
    - `@demo_demoinv_CustomerTYPES`, `@demo_demoinv_Customer`
    - `@demo_sync_invoke`
3. At the second SQL prompt, connect as bar/manager.
4. Run the following SQL scripts:

- `@demo_demoimp_CustomerTYPES`
- `@demo_demoimp_Customer`

5. Start the demoinv and demoimp adapters using the start scripts.
6. In invoke side SQL prompt, run `exec newCustomer_sync(id, city, state, zip, timeout)`.

   A new row in the `customers` table in `foo` schema is created. This new row has `Status` initially set to `None` but changes to `Success` when the invoking adapter receives a response from the implementing adapter.

   A new row is also created in the `results` table in `bar` schema. If the invoking adapter does not receive a response within the time specified in seconds, in the `timeout` parameter, then the `Status` column is not updated in `foo.customers`; instead, a new row is created in the correlation table `cus_newcustomer_`*`repo_owner_version`*. This table is created by the iStudio exported PL/SQL code. If necessary, `foo.customers` has a trigger to update automatically when a new row is created in the correlation table.

4.2.2 Related Files for Synchronous Invoke Implement

The following scripts are related to the runtime steps described in both cases in CASE TWO.

- `demo_sync_invoke.sql`

```
CREATE OR REPLACE PROCEDURE newCustomer_sync(
  ID NUMBER,
  CITY LONG,
  STATE LONG,
  ZIP LONG,
  timeout NUMBER)
AS
  moid NUMBER;
  aoid NUMBER;
  addrid NUMBER;
  corrid NUMBER;
  ret_id NUMBER;
  ret_status LONG;
BEGIN
  insert into customers values (id, Address_Array(Address(city, state, zip)),
                              'NONE');
  Customer.crMsg_newCustomer_repo_owner_version(moid, aoid, id);
  addrid := Customer.cr_ADDRESS_ARRAY_ADDRESS(city, state, zip, moid, aoid);
  corrid := Customer.inv_newCustomer_repo_owner_version(moid, 'demoinv',
timeout,
                                               ret_id, ret_status);
  update customers set status=ret_status where id=ret_id;
  COMMIT;
END;
/
```

- `demo_setup.sql`

```
CREATE USER foo identified by manager;
GRANT connect, resource to foo;
CREATE USER bar identified by manager;
GRANT connect, resource to bar;
CREATE OR REPLACE TYPE foo.Address IS OBJECT (
city            VARCHAR2(1000),
state            VARCHAR2(1000),
zip            VARCHAR2(1000)
);
/
CREATE OR REPLACE TYPE foo.Address_Array IS VARRAY(1000) OF foo.Address;
/
CREATE TABLE foo.customers (id NUMBER, address foo.Address_Array, status
VARCHAR2(20));
CREATE OR REPLACE TYPE bar.Address IS OBJECT (
city            VARCHAR2(1000),
state            VARCHAR2(1000),
zip            VARCHAR2(1000)
);
/
CREATE OR REPLACE TYPE bar.Address_Array IS VARRAY(1000) OF bar.Address;
/
CREATE TABLE bar.results (id NUMBER, address bar.Address_Array);
```

4.2.3 Asynchronous Invoke and Implement

Run the `demo_setup.sql` file to create necessary schemas in the database on the application or spoke database. It may be necessary to connect as the `system` user.

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

4.2.3.1 Design Time Steps

1. Create a Business Object in iStudio. Enter Customer in the Business Object Name field in the Create Business Object dialog.
2. Create a common data type.
3. Create a procedure in iStudio. In the Create Procedure dialog, complete the following:
   a. Select Customer for the Business Object.
   b. Enter newCustomer in the Procedure Name field.
   c. Click **Import** and select **Database** to import attributes from the database.
   d. Log in to the Database using the correct information.
      * Expand the FOO schema, Tables/Views, and select `FOO.CUSTOMERS`.
      * In the right hand side of the dialog, select the ID, ADDRESS, and STATUS columns using the control key.
      * Click **Done**.
      * Import arguments as IN arguments. Change the last column (IN/OUT/INOUT) for Status to Out and click **Save**.
4. Create an application in iStudio. Enter demoinv in the Application Name field in the Create Application dialog
5. Create an invoked procedure using the Invoke Wizard in iStudio:
   a. Select demoinv for the Application and Database as the Message Type in the Select a Procedure page.
   b. Expand the list in the Select a Procedure page and select newCustomer.

   c. Click **Import** and select Common View in the Define Application View page to import attributes from the common view.

   d. Change the `ID` attribute from `IN` to `INOUT`.

      See Also: Appendix A, "Frequently Asked Questions"

   e. Uncheck the box for Synchronous.

   f. Click **Returned In Args** and enter the following:

      * In Argument: ID
      * Out Argument: ID

6. Create the following mapping for the newCustomer procedure in the Define Mapping IN Arguments page:

   ■ `newCustomer:IN [demoinv View] -- Object Copy -- newCustomer:IN [Common View]`

7. Create the following mapping for the newCustomer procedure in the Define Mapping OUT Arguments page:

   ■ newCustomer:OUT.STATUS [Common View] -- Copy Fields -- newCustomer:OUT.STATUS [demoinv View]

8. Edit the SQL code on the Define Stored Procedure page as follows:

   ■ For `sub_newCustomer_`*`repo_owner_version`*, following the line `dummy:= 0;`, enter update customers set status=`sub_newCustomer_`*`repo_owner_version`*.status where id=`sub_newCustomer_`*`repo_owner_version`*;

9. Click **Finish**.

10. Create a second application in iStudio. Enter demoimp in the Application Name field in the Create Application dialog.

11. Create an implemented procedure using the Implement Wizard in iStudio:

   a. Select demoimp for the Application and Database as the Message Type.

   b. Expand the list in the Select a Procedure page and select newCustomer.

   c. Click **Import** and select Database in the Define Application View page to import attributes from the database.

   d. Enter the correct information in the Database Login dialog.

      * Expand BAR, Tables/Views, and select `BAR.RESULTS`.
      * In the right hand side of the dialog, select the ID, ADDRESS, and STATUS columns using the control key.
      * Click **Done**.
      * Import arguments as IN arguments. Add an attribute called STATUS [String, OUT].

12. Create the following mapping for the newCustomer procedure in the Define Mapping IN Arguments page:

   ■ `newCustomer:IN [Common View] -- Object Copy -- newCustomer:IN [demoimp View]`

13. Create the following mapping for the newCustomer procedure in the Define Mapping OUT Arguments page:

- `newCustomer:OUT [dempimp View] -- Object Copy -- newCustomer:OUT [Common View]`

14. Edit the SQL code in the Define Stored Procedure page as follows:
    - For `imp_newCustomer_`*`repo_owner_version`*, following the line `dummy:= 0;`, enter `insert into results values(i_id, i_address);o_status:= 'SUCCESS';`
15. Click **Finish**.
16. To Export SQL code, right-click **Applications** in iStudio, and select **Export PL/SQL**. Select demoinv and demoimp from the context menu.
17. Enter demo for the File Prefix field.

    The following files are created and stored in the *`ORACLE_HOME`*`/integration/interconnect/iStudio` directory:
    - `demo_demopub_Customer.sql`
    - `demo_demopub_CustomerTYPES.sql`
    - `demo_demosub_Customer.sql`
    - `demo_demosub_CustomerTYPES.sql`

4.2.3.2 Runtime Steps

Bring up two SQL prompts:

1. At the first SQL prompt, connect as foo/manager.
2. Run the following SQL scripts:
   - `@demo_demoinv_CustomerTYPES`
   - `@demo_demoinv_Customer`
   - `@demo_invoke`.
3. At the second SQL prompt, connect as bar/manager.
4. Run the following SQL scripts:
   - `@demo_demoimp_CustomerTYPES`
   - `@demo_demoimp_Customer`.
5. Start the demoinv and demoimp adapters.
6. In invoke side SQL prompt, run `exec newCustomer_async(id, city, state, zip, timeout)`.

   A new row is created in the `customers` table in the `demoinv` schema. This new row has `STATUS` initially set to `none` but changes to `success` if the invoking adapter receives a response from the implementing adapter. A new row is created in the `Results` table in the `bar` schema.

4.2.4 Related Files for Asynchronous Invoke and Implement

The following scripts are related to the runtime steps described asynchronous invoke/implement:

- `demo_async_invoke.sql`

```
CREATE OR REPLACE PROCEDURE newCustomer_async(
  ID NUMBER,
  CITY LONG,
  STATE LONG,
  ZIP LONG)
AS
  moid NUMBER;
  aoid NUMBER;
  addrid NUMBER;
BEGIN
  insert into customers values (id, Address_Array(Address(city, state, zip)),
                              'NONE');
  Customer.crMsg_newCustomer_repo_owner_version(moid, aoid, id);
  addrid := Customer.cr_ADDRESS_ARRAY_ADDRESS(city, state, zip, moid, aoid);
  Customer.inv_newCustomer_repo_owner_version(moid, 'demoinv');
  COMMIT;
END;
/
```

- demo_setup.sql

```
CREATE USER foo identified by manager;
GRANT connect, resource to foo;
CREATE USER bar identified by manager;
GRANT connect, resource to bar;
CREATE OR REPLACE TYPE foo.Address IS OBJECT (
city             VARCHAR2(1000),
state             VARCHAR2(1000),
zip             VARCHAR2(1000)
);
/
CREATE OR REPLACE TYPE foo.Address_Array IS VARRAY(1000) OF foo.Address;
/
CREATE TABLE foo.customers (id NUMBER, address foo.Address_Array, status
VARCHAR2(20));
CREATE OR REPLACE TYPE bar.Address IS OBJECT (
city             VARCHAR2(1000),
state             VARCHAR2(1000),
zip             VARCHAR2(1000)
);
/
CREATE OR REPLACE TYPE bar.Address_Array IS VARRAY(1000) OF bar.Address;
/
CREATE TABLE bar.results (id NUMBER, address bar.Address_Array);
```

A
Frequently Asked Questions

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

- What should I enter on the Database User Configuration screen during installation?
- Is it possible to edit the database configuration settings created during installation?
- How can I specify a listener port other than 1521?
- Can I install multiple Database adapters on the same computer?
- If we manually deploy the PL/SQL code, where is the code, exported through iStudio, saved?
- What is the Returned IN Args feature in iStudio and how do I use it?
- How do I deploy PL/SQL code to use with the Database adapter?
- Can database messages contain arrays of arrays?
- When I run start, I do not view anything happening - no log files are created and I don't view any messages in the console - how do I get back to the command prompt?
- Why do I get errors when trying to load PL/SQL code generated through iStudio?
- What are the steps to prepare a Database adapter that publishes events?
- What are the steps to prepare a Database adapter that invokes procedures?
- What are the steps to prepare a Database adapter that subscribes to events?
- What are the steps to prepare a Database adapter that implements procedures?
- How do I secure my passwords?

What should I enter on the Database User Configuration screen during installation?

This information is used to find where the stored procedures generated through iStudio will be installed for application inbound messages. At runtime, the Database adapter uses this information to call a user-specified stored procedure. This user can be an existing user or a user created specifically for OracleAS Integration InterConnect.

Is it possible to edit the database configuration settings created during installation?

Edit the `adapter.ini` file located in the `ORACLE_HOME/integration/interconnect/adapters/[AppType][Partition]` directory.

> **See Also:** Chapter 2, "Installation and Configuration"

How can I specify a listener port other than 1521?

Edit the `db_bridge_schema#_port` parameter.

> **See Also:** Chapter 2, "Installation and Configuration"

Can I install multiple Database adapters on the same computer?

Using the Oracle Universal Installer, only one Database adapter can be installed in a single Oracle home. However, copies of the Database adapter using the `copyAdapter` script available in the `ORACLE_HOME/integration/interconnect/bin` directory. Usage: `copyAdapter dbapp1 dbapp2`

The script will create a copy of the already installed Database adapter called dbapp1 with a name of dbapp2.

If we manually deploy the PL/SQL code, where is the code, exported through iStudio, saved?

The PL/SQL code is saved in the `ORACLE_HOME/integration/interconnect/iStudio` directory. iStudio allows any extension to be specified, which is used to prefix the name of every SQL file, generated through iStudio. The following convention is used in naming the SQL files:

```
PrefixSpecifiedInIStudio_ApplicationName_BusinessObjectTYPES.sql
PrefixSpecifiedInIStudio_ApplicationName_BusinessObject.sql
```

What is the Returned IN Args feature in iStudio and how do I use it?

Please refer to "Returned In Arguments" on page 3-4.

How do I deploy PL/SQL code to use with the Database adapter?

The following steps describe how to deploy PL/SQL code for the Database adapter:

1. Click the **Deploy** tab in the iStudio window.
2. Right-click a Database application and select **Deploy PL/SQL**. The Deploy PL/SQL - Select Events/Procedures screen is displayed.
3. Select the application, event or procedure to deploy the corresponding PL/SQL.
4. Click **Next**. The Deploy PL/SQL - Database Information screen is displayed. This page allows you to specify the database connection information for deploying the PL/SQL code.
5. Enter information in the following fields:
   – Database username: The database user name required for connecting to the database.
   – Database password: The password required for connecting to the database.

   - Database URL: The URL of the database required for connecting to the database. The URL should be in the form: `host:port:SID`.

6. Click **Next**. The Deploy PL/SQL - Summary screen is displayed, which displays a summary of the database connectivity information entered in the previous screen.
7. The Deploy PL/SQL - Summary screen displays the following:
   - Database Information
   - Selected Events/Procedures

     This page displays a list of selected packages and the corresponding procedures contained in those packages that you have selected for deployment. The status of each package appears in parenthesis next to the package name.
8. Click **Next**. The Deploy PL/SQL - Status screen is displayed.
9. Click **Deploy**. The generated PL/SQL is deployed for the selected application, event or procedure.

If you do not want to export all stored procedures, for all applications, as this can take a while, select one or more applications. Only the stored procedures for those applications will be generated. You can also select messages based on the role; for example, if you select publish, then only publish messages will be generated. Or, you can choose to export the stored procedures for specific messages by selecting those messages in the list.

Can database messages contain arrays of arrays?

The database does not allow arrays of arrays. Thus, the application view of database messages should not contain arrays of arrays. For example, the application view of an database message can contain an array of Customers, where each message contains one Address. However, it cannot contain an array of Customers, where each contains an array of Addresses.

When I run start, I do not view anything happening - no log files are created and I don't view any messages in the console - how do I get back to the command prompt?

A start executable that is not the OracleAS Integration InterConnect `start` script must be running. This is dependent on what is in the PATH environment variable. Thus, run the `start` script as follows:

Platform	Executable
UNIX	`./start`
Windows	Use the Service Panel.

Why do I get errors when trying to load PL/SQL code generated through iStudio?

Ensure you none of the PL/SQL reserved keywords are used in OracleAS Integration InterConnect messages. For example, for a Phone object contains the attributes `areacode` and `number`, a problem would occur because `number` is a reserved keyword in PL/SQL.

What are the steps to prepare a Database adapter that publishes events?

Before a Database adapter can publish events, some stored procedures need to be generated in iStudio.

iStudio will create two SQL scripts for a publish message; one with stored procedures and one with types. The `types` script name will end with `TYPES.sql`. Using any user name, load the `types` scripts and the stored procedure script into the database.

When an event occurs, there are several PL/SQL methods that must be called to publish the event message. All of the methods reside in the *`event business object`* package which is created in the stored procedure SQL script. The first procedure that must be called is `crMsg_`*`event name`*`_`*`event owner`*`_`*`event version`*. It has two out arguments which are both of type `number`: the message id and the root data type id.

Next, populate the message with the correct data. For each non-primitive attribute that the message contains, there is a function called `cr_`*`data type name`*`_`*`attribute name`*. This function has one argument for each primitive attribute it contains and it takes the message id and the parent data type id. It returns a number, which is the data type id. When all data types have been created, a procedure must be called to publish the message. This procedure is named `pub_`*`event name`*`_`*`eventowner`*`_`*`event version`*. This procedure has three arguments: the message id, the source application name, and the destination application name. The destination application name is ignored, so pass in whatever is applicable.

For example, an event in the `Customer` business object is called `create`. Application `A` publishes this event. The application view of this event contains an attribute called `C` of type `cust`. The `cust` type contains a `name` attribute, which is a String and a `loc` attribute of type `Location`. The `Location` type contains a `city` attribute, which is a String, and a `state` attribute, which is also a String. The following piece of code would publish a `create` event.

```
DECLARE
  moid NUMBER;
  aoid NUMBER;
  custid NUMBER;
  locid NUMBER;
BEGIN
  Customer.crMsg_create_TEST_V1(moid, aoid);
  custid := Customer.cr_cust_c('Homer', moid, aoid);
  locid := Customer.cr_Location_loc('Redwood Shores', 'CA', moid, custid);
  Customer.pub_create_TEST_V1(moid, 'a', '');
END
```

What are the steps to prepare a Database adapter that invokes procedures?

This is very similar to publishing events. All of the steps are the same until the final procedure call. The name is `inv_`*`proc name`*`_`*`proc_owner`*`_`*`proc version`* and has three IN arguments: the message id, the source application name, and a timeout. The timeout is how many seconds to wait for a response. The event also has as many OUT arguments as the procedure defined in iStudio has.

What are the steps to prepare a Database adapter that subscribes to events?

Before a Database adapter can subscribe to events, some stored procedures need to be generated in iStudio.

iStudio will create two SQL scripts for a subscribe message: one with stored procedures and one with types. The types script name will end with `TYPES.sql`. Under the same user name specified on the Database Configuration page during installation, load the `types` scripts and the stored procedure script into the database. A pre-existing user can be specified, but if a user name that does not exist is entered, that user must be created manually.

The DB adapter will call the procedure `sub_`*`event name_event owner_event version`* in the package *`eventbusiness object`* when a message is received. Add PL/SQL code in this method to perform whatever tasks are necessary when this kind of message is received. This code can be added in iStudio when creating the message, or modify the stored procedure SQL script before loading it into the database.

What are the steps to prepare a Database adapter that implements procedures?

The steps are very similar to subscribing to events. However, the procedure that the Database adapter will call is `imp_`*`procname_proc owner_proc version`*. This procedure will have OUT arguments corresponding to the OUT arguments in the procedure defined in iStudio. In addition to writing PL/SQL code to perform the necessary tasks, the OUT arguments must be filled in with correct values. Write this code in iStudio when creating the message, or modify the stored procedure SQL script before loading it into the database. If the `start` script is used to start the Database adapter, there is a way to determine whether the Database adapter was started properly. This can be viewed in the `oailog.txt` file in the logs directory of the Database adapter.

How do I secure my passwords?

OracleAS Integration InterConnect uses Oracle Wallet Manager to maintain system passwords. When you install OracleAS Integration InterConnect, Oracle Wallet Manager is also installed and a password store is created. All passwords used by OracleAS Integration InterConnect components are stored in the password store. The password is stored in the Oracle Wallet in the following format:

```
ApplicationName/password
```

For example,

```
AQAPP/aq_bridge_schema_password
```

The `ApplicationName` is the name of the application, which is extracted from the `adapter.ini` file of the corresponding adapter. In the `adapter.ini` file, the `application` parameter specifies the `ApplicationName` to which this adapter connects. The password for the application is also retrieved from the `adapter.ini` file.

You can create, update, and delete passwords using the `oraclewallet` command. When you run the command, it prompts you for the admin password.

You can use the following commands to manage your passwords:

- List all passwords in the store

  ```
  oraclewallet -listsecrets
  ```

- Create a password

  ```
  oraclewallet -createsecret passwordname
  ```

  For example, to create a password for the hub schema:

```
oraclewallet -createsecret hub_password
```

- View a password

  ```
  oraclewallet -viewsecret passwordname
  ```

  For example, to view the password for the hub schema:

  ```
  oraclewallet -viewsecret hub_password
  ```

- Update a password

  ```
  oraclewallet -updatesecret passwordname
  ```

  For example, to update the password for the hub schema:

  ```
  oraclewallet -updatesecret hub_password
  ```

- Delete a password

  ```
  oraclewallet -deletesecret passwordname
  ```

  For example, to delete the password for the hub schema:

  ```
  oraclewallet -deletesecret hub_password
  ```

Index

A

Advanced Queuing payload, 3-1
advanced queuing payload, 3-4
agent_admin_port, 2-5
agent_delete_file_cache_at_startup, 2-5
agent_dvm_table_caching, 2-5
agent_log_level, 2-5
agent_lookup_table_caching, 2-6
agent_max_ao_cache_size, 2-6
agent_max_co_cache_size, 2-6
agent_max_dvm_table_cache_size, 2-6
agent_max_lookup_table_cache_size, 2-6
agent_max_message_metadata_cache_size, 2-6
agent_max_queue_size, 2-6
agent_message_selector, 2-6
agent_metadata_caching, 2-6
agent_persistence_cleanup_interval, 2-7
agent_persistence_queue_size, 2-7
agent_persistence_retry_interval, 2-7
agent_pipeline_from_hub, 2-7
agent_pipeline_to_hub, 2-7
agent_reply_message_selector, 2-7
agent_reply_subscriber_name, 2-7
agent_subscriber_name, 2-7
agent_throughput_measurement_enabled, 2-8
agent_tracking_enabled, 2-8
agent_use_custom_hub_dtd, 2-8
application, 2-8
application parameter, 2-4

C

configuration, 2-3
  adapter.ini, 2-5
  ini file settings, 2-4

D

database adapter
  configuration, 2-3
  design time concepts, 3-1
  hardware requirements, 1-1
  how it works, 3-5
  importing database tables, 3-1
  installation, 2-1
  installation steps, 2-1
  overview, 1-1
  preinstallation tasks, 2-1
  runtime concepts, 3-5
  software requirements, 1-1
  starting, 3-5
  stopping, 3-6
  verification test, 2-3
Database Adapter-specific Parameters, 2-12
Database Requirements, 1-2
database tables
  importing, 3-1
db_bridge_schema#_num_writers, 2-12
db_bridge_schema#_password, 2-12
db_bridge_schema#_port, 2-13
db_bridge_schema#_username, 2-13
db_bridge_schema#_writer_password, 2-13
db_bridge_schema#_writer_use_oracle_objects, 2-13
db_bridge_schema#_writer_username, 2-13
db_bridge_sql_ trace, 2-13
db_bridge_use_thin_jdbc, 2-13
design time concepts, 3-1

E

EAI, 1-1
encoding, 2-8
Enterprise Application Integration, 1-1
export
  pl/sql code, 3-4

H

How do I secure my passwords?, A-5
hub_host, 2-4
hub_hostx, 2-5
hub_instance, 2-4
hub_instancex, 2-5
hub_num_nodes, 2-5
hub_port, 2-4
hub_portx, 2-5
hub_username, 2-4
Hub.ini Parameters, 2-4

I

import
  oracle object, advanced queuing payload, 3-4
installation, 2-1
  preinstallation, 2-1
  verification test, 2-3
instance_number, 2-8

J

JRE Requirements, 1-2

L

Log File of Database adapter, 3-6
logs, 2-3

N

nls_country, 2-9
nls_date_format, 2-9
nls_language, 2-9

O

Object, 3-1
Operating System Requirements, 1-2
Oracle Object, 3-1
oracle object, 3-4

P

partition, 2-9
persistence, 2-3
pl/sql code, 3-4

R

RAC-specific adapter.ini parameters, 2-13
Real Application Clusters Hub.ini Parameters, 2-4
receiving adapter, 3-5
Relational, 3-1
repository_name, 2-4
returned in arguments, 3-4
runtime concepts, 3-5

S

sample use cases
  asynchronous invoke implement, 4-8
  invoke and implement, 4-4
  publish subscribe, 4-1
  synchronous invoke implement, 4-4
sender adapter, 3-5
service_class, 2-9
service_classpath, 2-9
service_jdk_dll, 2-10
service_jdk_version, 2-10
service_max_heap_size, 2-10
service_max_java_stack_size, 2-10
service_max_native_stack_size, 2-10
service_min_heap_size, 2-10
service_num_vm_args, 2-11
service_path, 2-11
service_vm_argnumber, 2-11