Oracle® Application Server Integration InterConnect

Adapter for DB Installation and User's Guide

10*g* Release 2 (10.1.2)

B14076-02

December 2005

ORACLE®

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

B14076-02

Primary Author: Rima Dave

Contributing Author: Vimmika Dinesh

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

Contents

Preface

This Preface contains these topics:

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

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

Accessibility of Code Examples in Documentation

Screen readers 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, some screen readers 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.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

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*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

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 run-time 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 installing the Database adapter.

Table 1–1 Hardware Requirements

Hardware	Windows 2000	UNIX
Disk Space	400 MB	400 MB
Memory	512 MB	512 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 installing the Database adapter.

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, then 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:

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:

2.1.1 Preinstallation Tasks

Refer to 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 about 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 screen of the OracleAS Integration InterConnect installation, select OracleAS Integration InterConnect Adapter For Database 10.1.2.0.2, 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 screen is displayed.
5. Enter the application name. Blank spaces are not permitted. The default value is `myDBApp`.

   Note: You can change the application name in iStudio after installation. In such a case, you need to specify the password corresponding to new application name in the Oracle Wallet.

   For more informtion, refer to the following sections in Appendix A, "Frequently Asked Questions":

   - Section A.17, "My Database adapter is not starting. What could be the reason?"
   - Section A.2, "How do I secure my passwords?"

   ---

6. Click **Next**. The Specify Spoke Database Connection Information screen 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 screen is displayed.
9. Enter and confirm the password for the bridge schema on the screen.
10. Click **Next**. The Summary screen 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 Installing Multiple Database Adapters in the Same Oracle Home

To install multiple instances of the Database adapter in same Oracle home, use the `copyAdapter` script located in the *ORACLE_HOME*/integration/interconnect/bin directory.

Usage: `copyAdapter` *app1 app2*

For example, you have one instance of Database adapter with name `myDBApp` installed on a computer. To install another instance of the Database adapter with name `myDBApp1` in the same Oracle home, use the following command:

```
copyAdapter myDBApp myDBApp1
```

The `copyAdapter` script is copied to the following `bin` directory only during Hub installation:

- UNIX: `ORACLE_HOME/integration/interconnect/bin`
- Windows: `ORACLE_HOME\integration\interconnect\bin`

If you need to use this script to create multiple adapters on a spoke computer, then copy the script to the `bin` directory on the spoke computer, and edit the script to reflect the new Oracle home.

After running the `copyAdapter` script, If you want to manage or monitor the newly installed adapter through Oracle Enterprise Manager 10*g* Application Server Control Console, then you need to modify the `opmn.xml` file by adding information about the new instance. For example, you have created a new instance of the Database adapter `myDBApp1` by using the `copyAdapter` script. To manage the `myDBApp1` adapter through Enterprise Manager, perform the following:

1. Navigate to the *MiddleTier*\bin directory and run the following command to stop the Enterprise Manager:

   ```
   emctl stop iasconsole
   ```

2. Next, specify the information about this new instance in the `opmn.xml` file located in the *ORACLEMIDDLETIER_HOME*/opmn/conf directory as follows:

   ```
   <process-type id="myDBApp1" module-id="adapter" working-dir="$ORACLE_
   ```

```
HOME/integration/interconnect/adapters/myDBApp1" status="enabled">
       <start timeout="600" retry="2"/>
       <stop timeout="120"/>
       <port id="icadapter_dmsport_range" range="15701-15800"/>
       <process-set id="myDBApp1" restart-on-death="true" numprocs="1">
            <module-data>
                <category id="start-parameters">
                    <data id="java-parameters" value="-Xms8M"/>
                    <data id="class-name"
                     value="oracle.oai.agent.service.AgentService"/>
                </category>
                <category id="stop-parameters">
                    <data id="java-parameters" value="-mx64m"/>
                    <data id="class-name"
                     value="oracle.oai.agent.proxy.ShutdownAgent"/>
                    <data id="application-parameters"
                     value="persistence/Agent.ior"/>
                </category>
            </module-data>
       </process-set>
</process-type>
```

The `opmn.xml` file would appear like this:

```
<process-type id="myDBApp" module-id="adapter" working-dir="$ORACLE
_HOME/integration/interconnect/adapters/myDBApp" status="enabled">
       <start timeout="600" retry="2"/>
       <stop timeout="120"/>
       <port id="icadapter_dmsport_range" range="15701-15800"/>
       <process-set id="myDBApp" restart-on-death="true" numprocs="1">
            <module-data>
                <category id="start-parameters">
                    <data id="java-parameters" value="-Xms8M"/>
                    <data id="class-name"
                     value="oracle.oai.agent.service.AgentService"/>
                </category>
                <category id="stop-parameters">
                    <data id="java-parameters" value="-mx64m"/>
                    <data id="class-name"
             value="oracle.oai.agent.proxy.ShutdownAgent"/>
             <data id="application-parameters"
              value="persistence/Agent.ior"/>
         </category>
     </module-data>
  </process-set>
</process-type>

<process-type id="myDBApp1" module-id="adapter" working-dir="$ORACLE
_HOME/integration/interconnect/adapters/myDBApp1" status="enabled">
       <start timeout="600" retry="2"/>
       <stop timeout="120"/>
       <port id="icadapter_dmsport_range" range="15701-15800"/>
       <process-set id="myDBApp1" restart-on-death="true" numprocs="1">
        <module-data>
          <category id="start-parameters">
              <data id="java-parameters" value="-Xms8M"/>
              <data id="class-name"
               value="oracle.oai.agent.service.AgentService"/>
          </category>
          <category id="stop-parameters">
               <data id="java-parameters" value="-mx64m"/>
```

```
                <data id="class-name"
                 value="oracle.oai.agent.proxy.ShutdownAgent"/>
                <data id="application-parameters"
                 value="persistence/Agent.ior"/>
              </category>
            </module-data>
          </process-set>
</process-type>
```

3. Save the `opmn.xml` file.
4. Navigate to the *`MiddleTier`*`\opmn\bin` directory and run the following command to reload the OPMN:

   ```
   opmnctl reload
   ```

5. You can start the `myDBApp1` adapter by using the following command

   ```
   opmnctl startproc ias-component="InterConnect" process-type="myDBApp1"
   ```

6. Navigate to the *`MiddleTier`*`\bin` directory and run the following command to start the Enterprise Manager:

   ```
   emctl start iasconsole
   ```

7. Login to the Oracle Enterprise Manager 10*g* Application Server Control Console to view and manage the newly installed or copied adapter. For information about how to use Oracle Enterprise Manager 10*g* Application Server Control Console , refer to the *Oracle Application Server Integration InterConnect User's Guide*

Note: While installing multiple adapters in the same computer, the copyadapter script does not create entries for the new adapter's password in the Oracle Wallet. You need to manually create a password for this new adapter using the Oracle Wallet Manager. To store the password in Oracle Wallet, use the following format:

```
ApplicationName/password
```

The number of entries is dependent on the type of adapter. For example, Database adapter needs two entries whereas AQ Adapter needs only one entry. For more information about how to manage your passwords in Oracle Wallet, refer to Section A.2, "How do I secure my passwords?" in Appendix A, "Frequently Asked Questions"

2.3 Configuring the Database Adapter

After an Database adapter installation, you can configure it according to your requirements. 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`

Table 2–1 (Cont.) Oracle9i Database Server Adapter Directory

Platform	Directory
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 `log.xml` log file.
`persistence`	The messages are persisted in this directory. Do not edit this directory or its files.

2.3.1 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 run-time 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.

2.3.2 Database Adapter Ini File Settings

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

- hub.ini Parameters
- adapter.ini Parameters

2.3.2.1 hub.ini Parameters

The Database adapter connects to the hub database by using parameters in the hub.ini file located in the hub directory. Table 2–5 gives a descriptions and an example of each parameter.

Table 2–5 hub.ini Parameters

Parameters	Description	Example
hub_host	The name of the computer hosting the hub database. There is no default value. The value is set during installation.	hub_host=mpscottpc
hub_instance	The SID of the hub database. There is no default value. The value is set during installation.	hub_instance=orcl
hub_port	The TNS listener port number for the hub database instance. There is no default value. The value is set during installation.	hub_port=1521
hub_username	The name of the hub database schema (or user name). There 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=InterConnectRepos itory

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, then 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

2.3.2.2 adapter.ini Parameters

The Database adapter connects to the spoke application using parameters in the adapter.ini file. Table 2–7 gives a descriptions and an example of each parameter.

Table 2–7 adapter.ini Parameters

Parameter	Description	Example
agent_admin_port	Specifies the port through which the adapter can be accessed through firewalls. Possible value: A valid port number Default value: None	agent_admin_port=1059
agent_delete_file_cache_at_startup	Specifies whether to delete the cached metadata during 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
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

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

Parameter	Description	Example
`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 to which 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 (such as `'%,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`
`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`

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

Parameter	Description	Example
agent_pipeline_from_hub	Specifies whether to activate 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 activate 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
agent_throughput_measurement_enabled	Specifies if the throughput measurement is enabled. Set this parameter to true to activate 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

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

Parameter	Description	Example
`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:\ORACLEHOME\Integration\InterConnect\adapters\AQApp\`
`instance_number`	Specifies the instance number to which this adapter corresponds. Specify a value only if you have multiple adapter instances for the given application with the given partition. Possible value: An integer greater than or equal to `1` Default value: None	`instance_number=1`
`nls_country`	Specifies the ISO country code. The codes are defined by ISO-3166. Possible value: A valid code. A full list of the codes is available at http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html Default value: `US` **Note**: This parameter specifies date format and is applicable only for the date format.	`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`

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

Parameter	Description	Example
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 only for the date format.	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 Java Virtual Machine(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 only for Microsoft Windows.	service_classpath=D:\oracle\oraic\integration\interconnect\lib\oai.jar;D:\oracle\oraic\jdbc\classes12.zip
service_jdk_dll	Specifies the Dynamic Link Library (DLL) that the adapter JVM should use. Possible value: A valid jvm.dll Default value: jvm.dll This parameter is only for Microsoft Windows.	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 only for Microsoft Windows.	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 only for Microsoft Windows.	service_max_heap_size=536870912
service_max_java_stack_size	Specifies the maximum size to which the JVM stack can grow. Possible value: A valid JVM maximum stack size. Default value: Default value for the JVM. This parameter is only for Microsoft Windows.	service_max_java_stack_size=409600

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

Parameter	Description	Example
service_max_native_stack_size	Specifies the maximum size to which 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 only for Microsoft Windows.	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 only for Microsoft Windows	service_min_heap_size=536870912
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 only for Microsoft Windows.	service_num_vm_args=1
service_path	Specifies the environment variable PATH. The PATH variable is set before starting the JVM. Typically, list all directories that contain necessary DLLs. Possible value: The valid PATH environment variable setting. Default value: None This parameter is only for Microsoft Windows.	service_path=%JREHOME%\bin;D:\oracle\oraic\bin
service_vm_*argnumber*	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: Valid JVM arguments Default value: None This parameter is only for Microsoft Windows.	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

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

Letter	Description	Example
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
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

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

Parameter	Description	Example
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
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 Section A.2, "How do I secure my passwords?" in Appendix A, "Frequently Asked Questions" 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 Oracle database version 8.x and later releases. Set this to true unless using an Oracle database version 7.x. 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 Database 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_schema_hostx	Indicates host for the node x.	db_bridge_schema_host2= *mypc.us.oracle.com*
db_bridge_schema_instancex	Indicates instance on node x.	db_bridge_schema_instance2= *inst1*
db_bridge_schema_portx	Indicates port for node x.	db_bridge_schema_port2=*1521*

2.4 Uninstalling the Database Adapter

To uninstall the Database adapter, perform the following:

1. Navigate to the *MiddleTier*\opmn\bin directory.
2. Run the following command to check the adapter status.

   ```
   opmnctl status
   ```

3. If the Database adapter instance that you want to remove is running, stop it by using the the following command:

   ```
   opmnctl stopproc ias-component="InterConnect" process-type="DBApp"
   ```

   where HTTPApp is the name of the Database adapter instance.

4. Navigate to the *MiddleTier*\bin directory and run the following command to stop the Enterprise Manager:

   ```
   emctl stop iasconsole
   ```

5. Carefully, remove the adapter process-type entry from the opmn.xml file located in the *MiddleTier*\opmn\conf directory. For example, to remove an Database adapter instance DBApp1, delete the following information specific to the adapter instance:

   ```
   <process-type id="DBApp1" module-id="adapter" working-dir="$ORACLE_
   HOME/integration/interconnect/adapters/DBApp1" status="enabled">
          <start timeout="600" retry="2"/>
          <stop timeout="120"/>
          <port id="icadapter_dmsport_range" range="15701-15800"/>
          <process-set id="DBApp1" restart-on-death="true" numprocs="1">
               <module-data>
                   <category id="start-parameters">
                       <data id="java-parameters" value="-Xms8M"/>
                       <data id="class-name"
                        value="oracle.oai.agent.service.AgentService"/>
                   </category>
                   <category id="stop-parameters">
                       <data id="java-parameters" value="-mx64m"/>
                       <data id="class-name"
   ```

```
                    value="oracle.oai.agent.proxy.ShutdownAgent"/>
                  <data id="application-parameters"
                    value="persistence/Agent.ior"/>
                </category>
              </module-data>
        </process-set>
</process-type>
```

6. Save the `opmn.xml` file.
7. Navigate to the *`MiddleTier`*`\opmn\bin` directory and run the following command to reload the OPMN:

   ```
   opmnctl reload
   ```

8. Navigate to the *`ORACLE_HOME`*`\integration\interconnect\adapters` directory and delete the folder that was created for the removed adapter instance.
9. Navigate to the *`MiddleTier`*`\bin` directory and run the following command to start the Enterprise Manager:

   ```
   emctl start iasconsole
   ```

3

Design-Time and Run-Time Concepts

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

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:

3.1.1 Importing Database Tables and Objects

For a database application, the application and common views resemble the underlying database schema, so iStudio enables 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

Column	Data Type
NAME	VARCHAR2(200)
ID	NUMBER
ADDRESSES	LONG

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

Table 3–2 Customer

Attribute	Data Type
NAME	STRING
ID	INTEGER
ADDRESSES	STRING

When importing from a 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

Column	Data Type
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:

Column	Data Type
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

Attribute	Data Type
NAME	STRING
ID	INTEGER
ADDRESSES	ARRAY (marked as an ARRAY)

Where ADDRESS_ARRAY contains the following attributes:

Attribute	Data Type
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

Column	Data Type
NAME	VARCHAR2(200)
ID	NUMBER
ADDRESS	NUMBER (Foreign key)

Table 3–6 Address

Column	Data Type
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. The table appears as following:

Attribute	Data Type
ID	NUMBER
CITY	STRING
STATE	STRING
ZIP	NUMBER

2. Import the Customer table. The table appears as following:

Attribute	Data Type
NAME	STRING
ID	NUMBER
ADDRESS	NUMBER

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

Example 3–4 Oracle CLOB

Table 3–7 shows a simple object table with a column `Orders` of data type `CLOB`.

Table 3–7 Customer

Column	Data Type
ID	NUMBER
NAME	LONG
ORDERS	CLOB

When imported into iStudio, this table appears as shown in Table 3–8

Table 3–8 Customer

Attribute	Data Type
ID	DOUBLE
NAME	STRING
ORDERS	STRING

Example 3–5 Oracle Application Data Type

Table 3–9 shows a simple object table.

Table 3–9 Customer

Column	Data Type
CUSTOMER	CUST_TYPE
SOURCE	VARCHAR(50)

Where `CUSTOMER` column is of data type `CUST_TYPE`.

Table 3–10 CUST_TYPE

Column	Data Type
ID	NUMBER
NAME	VARCHAR(20)
ADDRESS	ADDR_TYPE

Where `ADDRESS` column is of data type `ADDR_TYPE`.

Table 3–11 ADDR_TYPE

Column	Data Type
STREET	VARCHAR(20)
CITY	VARCHAR(20
STATE	VARCHAR(20)

When imported into iStudio, this table appears as shown in Table 3–12.

Table 3–12 Customer

Attribute	Data Type
CUSTOMER	CUST_TYPE
ID	DOUBLE
NAME	STRING
ADDRESS	ADDR_TYPE
STREET	STRING
CITY	STRING
STATE	STRING
SOURCE	STRING

Example 3–6 Complex Data Type that includes Nested Table

Table 3–13 shows a simple object table with a column `Orders` of data type `CLOB`

Table 3–13 Business_Contacts

Column	Data Type
INDUSTRY	VARCHAR(20)
CUSTOMER_LIST	CUSTOMERS

where `Customer_List` column is of `Customers` data type.

Table 3–14 Customers Table

Column	Data Type
ID	DOUBLE
NAME	STRING
ADDRESS	ADDR_TYPE

where `ADDRESS` column is of `ADDR_TYPE` data type.

Table 3–15 ADDR_TYPE

Column	Data Type
STREET	STRING
CITY	STRING
STATE	STRING

When imported into iStudio, the `Business_Contacts` table appears as shown in Table 3–16.

Table 3–16 Business_Contacts

Column	Data Type
INDUSTRY	VARCHAR(20)
CUSTOMER_LIST	CUSTOMERS (array)
ID	DOUBLE

Table 3–16 (Cont.) Business_Contacts

Column	Data Type
NAME	STRING
ADDRESS	ADDR_TYPE
STREET	STRING
CITY	STRING
STATE	STRING

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 does not 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 box and the **select Out Arguments** option.

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 run time. 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 Run-Time Concepts

The following section describes the run-time 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 comprises of the database bridge and the run-time 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 or 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 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.

Note: The adapter subscribing to an event should be started before any other adapter can publish that event. If you publish an event before starting the subscribing adapter, then the event would not be delivered to the subscribing adapter.

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

3.3 Starting the Database Adapter

Based on the operating system, the process for starting 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.
  2. Select the ***OracleHomeOracleASInterConnectAdapter-Application*** service.
  3. Start the service based on the operating system.

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 `log.xml` files. The files are located in the time-stamped 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 started successfully.

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

  You can verify the stop status of the Database adapter by viewing the `log.xml` files. These files are located in the time-stamped 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 box.
2. Create a common data type. In the Create Data Type dialog box, 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 box, 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 box.

5. Create a Published Event using the Publish Wizard in iStudio:

   a. Select **demopub** from the Application list and **Database** from the Message Type list in the Select an Event dialog box.

   b. Expand the list in the Select an Event dialog box and select **createCustomer**.

   c. Click **Import** in the Define Application View dialog box to import attributes from the Common View.

   d. Create the following mapping for the `newCustomer` procedure on the Define Mapping IN Arguments dialog box:

      * `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 box.

7. Create a Subscribed Event using the Subscribe Wizard in iStudio.

   a. Select **demosub** from the Application list and **Database** from the Message Type list in the Select an Event dialog box.

   b. Expand the list in the Select an Event dialog box and select **createCustomer**.

   c. Click **Import** in the Define Application View dialog box 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 dialog box.

   e. Enter the following SQL code on the Define Stored Procedure dialog box:

      * 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 box, 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 Run-Time 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. Start 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. Start 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 run-time 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 box, 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 box, 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 box, 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 box.
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 dialog box.
   b. Expand the list in the Select a Procedure dialog box and select **newCustomer**.
   c. Click **Import** and select **Common View** on the Define Application View dialog box 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 dialog box:
   - `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 dialog box:
   - `newCustomer:OUT.STATUS [Common View] -- Copy Fields -- newCustomer:OUT.STATUS [demoinv View]`
8. In the Define Stored Procedure dialog box, do not edit the SQL code, it is correct.
9. Click **Finish**.
10. Create an application in iStudio. In the Create Application dialog box, 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 dialog box and select **newCustomer**.
    c. Click **Import** and select **Database** in the Define Application View dialog box to import attributes from the database.
    d. Enter the correct information on the Database Login dialog box for the BAR schema.
       * Expand BAR, Tables/Views and select `BAR.RESULTS`.
       * In the right hand side of the dialog box, 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 dialog box:
    - `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 dialog box:
    - `newCustomer:OUT [dempimp View] -- Object Copy -- newCustomer:OUT [Common View]`
14. Edit the SQL code in the Define Stored Procedure dialog box 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 Run-Time Steps

The run-time 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 run-time 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 box.
2. Create a common data type.
3. Create a procedure in iStudio. In the Create Procedure dialog box, 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 box, 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 box
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 dialog box.
   b. Expand the list in the Select a Procedure dialog box and select **newCustomer**.

c. Click **Import** and select **Common View** in the Define Application View dialog box 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 dialog box:

- `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 dialog box:

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

8. Edit the SQL code on the Define Stored Procedure dialog box 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 box.

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 dialog box and select **newCustomer**.

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

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

* Expand **BAR**, **Tables/Views**, and select **BAR.RESULTS**.
* In the right hand side of the dialog box, 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 dialog box:

- `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 dialog box:

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

14. Edit the SQL code in the Define Stored Procedure dialog box 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 Run-Time 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 run-time 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 the following frequently asked questions about the Database adapter:

A.1 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 run time, 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.

A.2 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
```

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.

The number of entries is dependent on the type of adapter. For example, Database adapter needs two entries whereas AQ Adapter needs only one entry. The following table lists the entries that will be created for each adapter:

Adapter	Entry In Oracle Wallet
AQ	*ApplicationName*/aq_bridge_password
HTTP	*ApplicationName*/http.sender.password
HTTP	*ApplicationName*/sender.wallet_password
SMTP	*ApplicationName*/smtp.receiver.password
MQ	*ApplicationName*/mq.default.password
FTP	*ApplicationName*/file.sender.password
FTP	*ApplicationName*/file.receiver.password
DB	*ApplicationName*/db_bridge_schema1_password
DB	*ApplicationName*/db_bridge_schema1_writer_password

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
  ```

A.3 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"

A.4 How can I specify a listener port other than 1521?

Edit the `db_bridge_schema#_port` parameter.

See Also: Chapter 2, "Installation and Configuration"

A.5 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 enables 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`

A.6 What is the Returned IN Args feature in iStudio and how do I use it?

Please refer to "Returned In Arguments" on page 3-6.

A.7 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 dialog box is displayed. This dialog box enables 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 dialog box is displayed, which displays a summary of the database connectivity information entered in the previous dialog box.
7. The Deploy PL/SQL - Summary dialog box displays the following:
   - Database Information
   - Selected Events/Procedures

     This dialog box 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 dialog box 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 select to export the stored procedures for specific messages by selecting those messages in the list.

A.8 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.

A.9 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.

A.10 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.

A.11 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
```

A.12 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.

A.13 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 Database 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.

A.14 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 `log.xml` file in the logs directory of the Database adapter.

A.15 How can I deliver a message to a specific partition of the publishing adapter?

The Database adapter polls the MESSAGEOBJECTTABLE table in the OAI schema to check for incoming messages and picks up any message that is meant for that adapter by checking the application name. If you have created partitions on an application and you want a particular partition to receive message from the MESSAGEOBJECTTABLE, then you need to specify the partition name along with the application name before putting the message into the MESSAGEOBJECTABLE.

When Database adapter is used to connect to an application, iStudio generates the required PL/SQL stored procedures for the events defined for the application in iStudio. These stored procedures enable an application to interface with OracleAS Integration InterConnect through the Oracle database. For example, Application `DBAPP` contains two partitions `PAR1` and `PAR2`. When you deploy a database event `Publish(purchase_order.CreatePO)` from iStudio, procedures `crMsg_Create_PO_OAI_V1` and `pub_Create_PO_OAI_V1` are generated automatically. To specify that the message should be delivered to the partition `PAR2`, create the following trigger that uses these generated procedures. Specify the application name followed by the partition name as value of the `srcAppName` parameter:

```
CREATE OR REPLACE TRIGGER DBAPP.NEW_PO
 BEFORE INSERT
 ON     DBAPP.PO_TABLE
 FOR    EACH ROW
DECLARE
 msg_id Number;
 ao_id Number;
BEGIN
dbapp.PURCHASE_ORDER.crMsg_Create_PO_OAI_V1(msg_id,ao_id,:new.price,
:new.quantity,:new.poid,:new.poitem);
dbapp.PURCHASE_ORDER.pub_Create_PO_OAI_V1(msg_id,'DBAPPPAR2');
END;
```

A.16 What SQL Data Types are Supported and Not Supported in iStudio?

The SQL data types supported in iStudio are:

- NUMBER
- DEC
- DECIMAL
- INTEGER
- NUMERIC
- INT
- REAL
- SMALLINT
- FLOAT
- DATE
- VARCHAR2
- LONG

- CHAR
- CLOB
- BLOB
- RAW
- DOUBLE
- All user-defined data types except arrays of arrays

The SQL data types not supported in iStudio are:

- NVARCHAR2
- BINARY_FLOAT
- BINARY_DOUBLE
- TIMESTAMP
- ROWID
- UROWID
- NCHAR
- NCLOB
- BFILE

A.17 My Database adapter is not starting. What could be the reason?

Following can be the reasons:

- Repository might not be running.
- Hub database might not be running.
- Spoke database might not be running.
- User information specified in `adapter.ini` file and in Oracle Wallet might not be correct.
- Oracle Wallet might not contain the password information corresponding to your application name. For example, during installation you defined the application name as `myDBApp`. Later, you changed the application name in iStudio to `DBApp`. In such case, you need to specify the password corresponding to the new application name `DBApp` in the Oracle Wallet. You can create password by using the `oraclewallet` command.

See Also: Section A.2, "How do I secure my passwords?"

A.18 Database adapter not publishing messages.

Perform the following:

- Check if the MESSAGEINFOTABLE in OAI schema has the message.
- Check if the trigger, that publishes the message to the OAI schema, has specified the application name similar to the one specified for the `application` parameter in `adapter.ini` file. If the partition paarmeter in the `adapter.ini` file has a value, then the trigger should specify the application name as the string concatenation of the values of `application` and `partition` properties.

- Ensure that while creating the message in the trigger, the parent-child relationships are properly handled in accordance with the structure created in iStudio
- Ensure that the value of the `db_bridge_schema1_num_readers` parameter in the `adapter.ini` file is more than one..

A.19 Database adapter not subscribing to messages.

Perform the following:

- Verify if you are seeing the log message "Subscribing to message (BusinessObject.EventName:versioninfo)" during the startup.
- Ensure that the value of the db_bridge_schema1_num_writers parameter in `adapter.ini` file is more than one.
- Ensure that you have deployed the sql from iStudio

Index

A

C

D

E

H

I

J

L

N

O

P

R

S

T

U