Oracle9*i*AS InterConnect Adapter for DB

Installation and User's Guide

Release 2 (9.0.2)

February 2002

Part No. A95447-01

ORACLE

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

Part No. A95447-01

Primary Author: Shashi Suravarapu

Contents

3 Design Time and Runtime Concepts

4 Sample Use Cases

5 Frequently Asked Questions

Index

Send Us Your Comments

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

Part No. A95447-01

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

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

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

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

  Oracle Corporation
  Oracle9*i* Application Server Documentation Manager
  500 Oracle Parkway, M/S 2op3
  Redwood Shores, CA 94065 USA

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

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

Preface

This preface contains these topics:

Audience

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

- install applications
- maintain applications

Documentation Accessibility

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

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

Organization

This document contains:

Chapter 1, "Introduction"

This chapter describes the 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 for the Database adapter.

Chapter 4, "Sample Use Cases"

This chapter provides sample use cases for the Database adapter.

Chapter 5, "Frequently Asked Questions"

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

Related Documentation

For more information, see these Oracle resources:

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

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

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

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

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

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

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

`http://otn.oracle.com/admin/account/membership.html`

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

`http://otn.oracle.com/docs/index.htm`

Conventions

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

- Conventions in Text

- Conventions in Code Examples
- Conventions for Microsoft Windows Operating Systems

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width font)`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width font)`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.

Convention	Meaning	Example
lowercase monospace (fixed-width font) italic	Lowercase monospace italic font represents placeholders or variables.	You can specify the *parallel_clause*. Run U*old_release*.SQL where *old_release* refers to the release you installed prior to upgrading.

Conventions in Code Examples

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

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

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

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`
. . .	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	`CREATE TABLE ... AS `*`subquery`*`;` `SELECT `*`col1`*`, `*`col2`*`, ... , `*`coln`*` FROM employees;`
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	`acctbal NUMBER(11,2);` `acct CONSTANT NUMBER(4) := 3;`

Convention	Meaning	Example
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. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

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

Conventions for Microsoft Windows Operating Systems

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

Convention	Meaning	Example
Choose Start >	How to start a program.	To start the Oracle Database Configuration Assistant, choose Start > Programs > Oracle - *HOME_NAME* > Configuration and Migration Tools > Database Configuration Assistant.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the file name begins with \\, then Windows assumes it uses the Universal Naming Convention.	`c:\winnt"\"system32` is the same as `C:\WINNT\SYSTEM32`
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`

	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	`C:\>exp scott/tiger TABLES=emp QUERY=\"WHERE job='SALESMAN' and sal<1600\"`
		`C:\>imp SYSTEM/`*`password`*
		`FROMUSER=scott TABLES=(emp, dept)`

HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*`HOME_NAME`*`TNSListener`

Convention	Meaning	Example
ORACLE_HOME and *ORACLE_BASE*	In releases prior to Oracle8*i* release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory that by default used one of the following names: ■ `C:\orant` for Windows NT ■ `C:\orawin95` for Windows 95 ■ `C:\orawin98` for Windows 98 This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle`. If you install Oracle9*i* release 1 (9.0.1) on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\ora90`. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle9i Database Getting Starting for Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\admin` directory.

1

Introduction

This chapter discusses the Database adapter and the hardware and software requirements. This chapter discusses the following topics:

- Database Adapter Overview

Database Adapter Overview

The Oracle9*i*AS InterConnect Database adapter enables an Oracle Application, typically PL/SQL-based, to be integrated with other applications using Oracle9*i*AS InterConnect. This adapter is useful in all enterprise application integration scenarios involving Oracle Database applications.

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

Hardware Requirements

The following table lists the hardware requirements for the computer on which the Database adapter is installed.

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

Software Requirements

The following are software requirements for the Database Adapter:

- Operating System Requirements
- JRE Requirements
- Database Requirements

Operating System Requirements

Table 1–1 lists operating system requirements for the computer on which the Database adapter is installed.

Table 1–1 Operating System Requirements

Operating System	Version
Windows NT/2000	version that supports at least 400 MHz
IBM AIX-Based Systems	version 4.3.3
Compaq Tru64 UNIX	version 5.0a or 5.1

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

JRE Requirements

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

Database Requirements

The Database adapter requires Oracle8*i* or later version of the database. Typically, the database should already be used by the application. If this database is not used by the application, install either the Oracle8*i* or 9*i* database.

2

Installation and Configuration

This chapter describes installation and configuration of the Database adapter. The following topics are discussed:

- Installing the Database Adapter
- Configuring the Database Adapter

Installing the Database Adapter

This section contains these topics:

Preinstallation Tasks

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

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

Consult the following guides before proceeding with Database adapter installation:

- *Oracle9i Application Server Installation Guide*, which includes information on:
  - CD-ROM mounting
  - Oracle Universal Installer startup
  - Oracle9*i*AS InterConnect installation
- *Oracle9iAS InterConnect User Guide*, which includes information on:
  - Oracle9*i*AS InterConnect software, hardware, and system requirements

---

Note: Oracle9*i*AS InterConnect Hub is installable through the Oracle9*i*AS InterConnect Hub installation type. You must install the Oracle9*i*AS InterConnect Hub before proceeding with the Database adapter installation.

Installation Tasks

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

1. On the Available Product Components page of the Oracle9*i*AS InterConnect installation, select Database adapter, then select Next.

   Consider the following scenarios:

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

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

   ---

2. If installing Oracle9*i*AS InterConnect for the first time on this machine, complete the following steps to enter the hub database information:

   a. On the Welcome page, select Next. The Database Configuration page displays. Enter information in the following fields:

      * Host Name—The host name of the machine where the hub database is installed.
      * Port Number—The TNS listener port for the hub database.
      * Database SID—The SID for the hub database.

   b. Click Next. The Database User Configuration page displays. Enter information in the following fields:

      * User Name—The hub database user name. Make sure the Oracle9*i*AS InterConnect Hub is installed. If the Hub is not installed, complete the installation and note the user name and password.
      * Password—The password for the hub database user.

3. Click Next. The Adapter Configuration page displays. Enter the application to be defined or already defined in iStudio in the Application Name field. White spaces or blank spaces are not permitted. The default value is `myDBApp`.

4. Select next. The Spoke Application Database Configuration page displays. Enter information in the following fields:

   - Host Name—The name of the machine where the application database is installed.
   - Port Number—The database TNS listener port.
   - Database SID—The SID for the application database.

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

5. Select Next. The Spoke Application Database User Name page displays. Enter information in the following fields:

   - User Name—The name the Database adapter uses to connect to the database.
   - Password—The password for the user name.

6. Select Next. Complete the fields for any other components selected for installation, such as other adapters. When finished, the Summary page displays.

7. Select Install to install the Database adapter and other selected components. The Database adapter is installed in the following directory:

Platform	Directory
Windows	`%ORACLE_HOME%\oai\9.0.2\adapters\`*`Application`*
UNIX	`$ORACLE_HOME/oai/9.0.2/adapters/`*`Application`*

Post-Installation Steps

After installation a set of post-installation steps displays. These steps are also copied to the `post-installation.txt` file in the Database adapter directory.

> **Note:** The default `sys` password is `change_on_install` unless otherwise specified during installation.

To create the schema used by the Database adapter, complete the following tasks:

1. Access the directory where the Database adapter is installed.
2. Execute the following for the correct database version:

Database Version	Code to Execute
Oracle7.x	`oaischema -create -pre8i sys/[password] [tnsname]`
Oracle8i	`oaischema -create -8i sys/[password] [tnsname]`
Oracle 9i	`oaischema -create sys/[password] [tnsname]`

where:

- Password—The password for the system user.
- tnsname—The tnsname for the spoke database.

Verification Test

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

Configuring the Database Adapter

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

Table 2–1 Database Adapter Adapter Directory

Platform	Directory
UNIX	`$ORACLE_HOME/oai/9.0.2/adapters/`*`Application`*
Windows	`%ORACLE_HOME%\oai\9.0.2\adapters\`*`Application`*

Table 2–2 Executable Files

File	Description
`start.bat` (Windows)	Takes no parameters, starts the adapter.
`start` (UNIX)	
`stop.bat` (Windows)	Takes no parameters; stops the adapter.
`stop` (UNIX)	
`ignoreErrors.bat` (Windows)	If an argument is specified, then the given error code will be ignored. If no argument is specified, than all error codes specified in the `ErrorCodes.ini` will be ignored.
`ignoreErrors` (UNIX)	

Table 2–3 Configuration Files

File	Description
`ErrorCodes.ini` (Windows and UNIX)	Should contain one error code per line.
`adapter.ini` (Windows and UNIX)	Consists of all the initialization parameters which the adapter reads at startup.

Table 2–4 Directories

File	Description
persistence	The messages are persisted in this directory. This directory or its contents should not edited
logs	The logging of adapter activity is done in subdirectories of the log directory. Each new run of the adapter creates a new subdirectory in which logging is done in an `oailog.txt` file.

Using the Application Parameter

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

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

Ini File Settings

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

Hub.ini

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

Parameter	Description	Example
`hub_username`	The name of the hub database schema (or username). The possible value is a valid hub database username. There is no default value.	`hub_username=myhub`
`hub_password`	The password for the hub database user. The possible value is a the valid password for the hub database user. There is no default value.	`hub_password=manager`
`hub_host`	The name of the machine hosting the hub database. The possible value is a the valid machine name. There is no default value.	`hub_host=mpjoshipc`
`hub_instance`	The valid SID of the hub database. The possible value is a valid SID. There is no default value.	`hub_instance=orcl`
`hub_port`	The TNS listener port number for the HUB database instance. The possible value is a TNS listener port number. There is no default value.	`hub_port=1521`
`repository_name`	The valid name of the repository this adapter talks to. The possible value is a valid repository name. There is no default value.	`repository_name=myrepo`

Adapter.ini

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

Parameter	Description	Example
`application`	The name of the application this adapter connects to. This must match with the name specified in iStudio during creating of metadata. Any alphanumeric string can be used. There is no default value.	`application=dbapp`
`partition`	The partition this adapter handles as specified in iStudio. Any alphanumeric string is a possible value. There is no default value.	`partition=germany`
`instance_number`	To have multiple adapter instances for the given application with the given partition, each adapter should have a unique instance number. Possible values are any integer greater than 1. There is no default value.	`instance_number=1`
`agent_log_level`	Specifies the amount of logging necessary. Possible values are: 0=errors only 1=status and errors 2=trace, status, and errors The default value is 1.	`agent_log_level=2`
`agent_ subscriber_name`	The subscriber name used when this adapter registers its subscription. The possible value is a valid Oracle Advanced Queuing subscriber name and there is no default value.	`agent_subscriber_ name=dbapp`
`agent_message_ selector`	Specifies conditions for message selection when registering its subscription with the hub. The possible value is a valid Oracle Advanced Queuing message selector string. There is no default value.	`agent_message_ selector=recipient_ list like '%,dbapp,%'`
`agent_reply_ subscriber_name`	The subscriber name used when multiple adapter instances for the given application with the given partition are used. Optional if there is only one instance running. The possible value is application name (`parameter: application`) concatenated with instance number (`parameter: instance_number`). There is no default value.	`If application=dbapp, instance_number=2, then, agent_reply_ subscriber_name=dbapp2`

Parameter	Description	Example
agent_reply_ message_selector	Used only if multiple adapter instances for the given application with the given partition. The possible value is a string built using concatenating application name (parameter:application) with instance number (parameter:instance_number). There is no default value.	If application=dbapp, instance_number=2, then agent_reply_ message_ selector=receipient_ list like'%,dbapp2,%'
agent_tracking_ enabled	Specifies if message tracking is enabled. Set to false to turn off all tracking of messages. Set to true to track messages with tracking fields set in iStudio. Possible values are true or false. The default value is true.	agent_tracking_ enabled=true
agent_ throughput_ measurement_ enabled	Specifies if throughput measurement is enabled. Set to true to turn on all throughput measurements. Possible values are true or false. The default value is true.	agent_throughput_ measurement_ enabled=true
agent_use_ custom_hub_dtd	Specifies if a custom DTD should be used for the common view message when handing it to the hub. By default adapters use an Oracle9*i*AS InterConnect-specific DTD for all messages sent to the hub as other Oracle9*i*AS InterConnect adapters will be retrieving the messages from the hub and know how to interpret them. Set to true if for every message, the DTD imported for the message of the common view is to be used instead of the Oracle9*i*AS InterConnect DTD. Only set to true if a Oracle9*i*AS InterConnect adapter is not receiving the messages from the hub. Possible values are true or false. There is no default value.	agent_use_custom_hub_ dtd=false
agent_metadata_ caching	Specifies the metadata caching algorithm. Possible values are: ■ startup—Cache everything at startup. This may take a while if there are a lot of tables in the repository. ■ demand—Cache metadata as it is used. ■ none—No caching. This slows down performance. The default value is demand.	agent_metadata_ caching=demand

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

Parameter	Description	Example
agent_max_dvm_table_cache_size	Specifies the maximum number of DVM tables to cache. Possible values are any integer greater than 1. The default value is 200.	agent_max_dvm_table_cache_size=200
agent_max_lookup_table_cache_size	Specifies the maximum number of lookup tables to cache. Possible values are any integer greater than 1. The default value is 200.	agent_max_lookup_table_cache_size=200
agent_max_queue_size	Specifies the maximum size that internal Oracle9*i*AS InterConnect message queues can grow. Possible values are any integer greater than 1. The default value is 1000.	agent_max_queue_size=1000
agent_persistence_queue_size	Specifies the maximum size that internal Oracle9*i*AS InterConnect persistence queues can grow. Possible values are any integer greater than 1. The default value is 1000.	agent_persistence_queue_size=1000
agent_persistence_cleanup_interval	Specifies how often the persistence cleaner thread should run. Possible values are any integer greater than 30000. The default value is 60000.	agent_persistence_cleanup_interval=60000
agent_persistence_retry_interval	Specifies how often the persistence thread should retry when it fails to push a Oracle9*i*AS InterConnect message. Possible values are any integer greater than 5000. The default value is 60000.	agent_persistence_retry_interval=60000
service_path	Windows only. The value that the environment variable PATH should be set to. path is set to the specified value before forking the Java VM. Typically, all directories containing all necessary DLLs should be listed here. Possible values are the valid path environment variable setting. There is no default value.	service_path=%JREHOME%\bin;D:\oracle\ora902\bin
service_classpath	The classpath used by the adapter Java VM. If a custom adapter is developed and as a result, the adapter is to be used to pick up any additional jars, add the jars to the existing set of jars being picked up. Possible values are the valid classpath. There is no default value.	service_classpath=D:\oracle\ora902\oai\902\lib\oai.jar;%JREHOME%\lib\i18n.jar;D:\oracle\ora902\jdbc\classes12.zip
service_class	The entry class for the Windows NT service. Possible values are oracle/oai/agent/service/AgentService. There is no default value.	service_class=oracle/oai/agent/service/AgentService
service_max_java_stack_size	Windows only. The maximum size to which the Java VM's stack can grow. Possible values are the valid Java VM maximum native stack size. The default value is the default for the Java VM.	service_max_java_stack_size=409600

Parameter	Description	Example
service_max_native_stack_size	Windows only. The maximum size to which the Java VM's native stack can grow. Possible values are the valid Java VM maximum native stack size. The default value is the default for the Java VM.	service_max_native_size=131072
service_min_heap_size	Windows only. Specifies the minimum heap size for the adapter Java VM. Possible values are the valid Java VM heap sizes. The default value is the default Java VM heap size.	service_min_heap_size=536870912
service_max_heap_size	Windows only. Specifies the maximum heap size for the adapter Java VM. Possible values are any valid Java VM heap sizes. The default value is 536870912.	service_max_heap_size=536870912
service_num_vm_args	Windows only. The number of service_vm_arg<number> parameters specified. Possible values are the number of service_vm_arg<number> parameters. There is no default value.	service_num_vm_args=1
service_vm_arg<number>	Windows only. Specifies any additional arguments to the Java VM. For example, to get line numbers in any of the stack traces, set service_vm_arg1=java.compiler=NONE. If there is a list of arguments to specify, use multiple parameters as shown in the example by incrementing the last digit starting with 1. Be sure to set the service_num_vm_args correctly. Possible values are any valid Java VM arguments. There is no default value.	service_vm_arg1=java.compiler=NONE service_vm_arg2=oai.adapter=.db
service_jdk_version	Windows only. The JDK version the adapter Java VM should use. The default value is 1.3.1.	service_jdk_version=1.3.1
service_jdk_dll	Windows only. The dll the adapter Java VM should use. The default value is jvm.dll.	service_jdk_dll=jvm.dll

Database Adapter Parameters

The following table lists the parameters specific to the Database adapter.

Parameter	Description	Example
bridge_class	Indicates the entry class for the Database adapter. Do not modify this value. The default value is oracle.oai.agent.adapter.db. DBBridge.	bridge_ class=oracle.oai.agent. adapter.db. DBBridge
db_bridge_use_ thin_jdbc	Indicates whether to use a thin JDBC driver when talking to the database. The possible values are true and false. The default is true.	db_bridge_thin_ jdbc=true
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, therefore enabling the SQL trace facility. For more information on the SQL trace facility, including how to format and interpret the output, see the Oracle Tuning Guide. The possible values are true or false; the default value is false.	db_bridge_sql_trace = true
db_bridge_ instance	The SID of the database instance. The possible value is the valid SID. There is no default value.	db_bridge_instance=orcl
db_bridge_num_ schemas	The number of schemas that this database adapter will connect to. The possible values are any integer greater than 0 and the default value is 1.	db_bridge_num_schemas = 1
db_bridge_ schema<#>_ username	The username for the schema number <schema#>. The possible values for the schema number are 1 through <db_bridge_num_schemas>. This username is used by all of the database readers corresponding to this schema. This corresponds to the user created by the post-install oaischema script. This value should not be modified. Possible values are any valid database user name and there is no default value.	db_bridge_schema1_ username=oai
db_bridge_ schema<#>_ password	The password for the user specified in the db_bridge_ schema<schema#>_username. The possible value is the password for the corresponding database user and There is no default value.	db_bridge_schema1_ password=oai encrypted_db_bridge_ schema1_ password=11251101106410 9110871093
db_bridge_ schema<#>_host	The name of the machine hosting the database instance specified by the db_bridge_schema<#>_instance. The possible value is the name of the machine hosting the database and There is no default value.	db_bridge_schema1_ host=ssuravar-sun

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. The possible value is any valid TNS listener port number and There is no default value.	db_bridge_schema1_ port=1521
db_bridge_ schema<#>_ instance	The SID of the database instance. The possible value is any valid SID and There is no default value.	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. The possible values are any integer greater than 0 and there is no default value.	db_bridge_schema1_num_ readers=1
db_bridge_ schema<#>_num_ writers	The number of database writers corresponding to the schema number. This is the same as the number of writer threads; each thread has its own database session. The possible values are any integer greater than 0 and there is no default value.	db_bridge_schema1_num_ writers=1
db_bridge_ schema<#>_ writer_username	The username to be used by this writer to log on to the database as specified by the db_bridge_schema<#>_ instance parameter. The possible values are any valid database user and there is no default value.	db_bridge_schema1_ writer_username=mydbapp
db_bridge_ schema<#>_ writer_password	The password corresponding to the database user specified by the db_bridge_schema<#>_writer_ username parameter. The possible values are any valid password and there is no default value.	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. The possible values are true or false and the default value is false.	db_bridge_schema1_ writer_use_oracle_ objects=true

3

Design Time and Runtime Concepts

This chapter describes the design time and runtime concepts for the Database adapter. Topics include:

Database Adapter Design Time Concepts

The following topics discuss the iStudio concepts pertinent to the Database adapter.

See Also: *Oracle9iAS InterConnect User Guide*

Importing Database Tables

For a database application, the application and common views resemble the underlying database schema. For this reason, iStudio allows the creation of a view by importing tables directly from the database. The following tables can be imported:

- Relational
- Object
- Oracle Object
- Advanced Queuing payload.

The following examples illustrate some basic features for the database tables.

Example: Relational Table

The following example illustrates basic features of a relational table.

Table 3–1 Customer

Parameter	Value
NAME	VARCHAR2(200)
ID	NUMBER
ADDRESSES	LONG

When imported into iStudio, this table results in the following:

Table 3–2 Customer

Parameter	Value
NAME	STRING
ID	INTEGER

Parameter	Value
ADDRESSES	STRING

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

Example: Object Table

The following example illustrates basic features of the 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 results in the following:

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

In this example, the hierarchical structure is kept intact when dealing with Oracle Object Types.

Example: FOREIGN Key

In an example similar to the previous one, the structure or relationship is represented using a FOREIGN key. In this case, the end user is responsible for importing each of the different tables and setting up the relationship in iStudio by editing the types of attributes. The following example illustrates this point.

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:

   ```
   CITY                STRING
   STATE               STRING
   ZIP                 NUMBER
   ```

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

   ```
   NAME                STRING
   ID                  NUMBER
   ADDRESS             NUMBER
   ```

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

Importing an Oracle Object or Advanced Queuing Payload

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.

Returned In Arguments

Returned In arguments applies when invoking procedures. The Returned IN Args page does only displays in the Invoke wizard. Returned In arguments are used to propagate `INOUT` attributes contained in the request to the reply. Without this feature, it would have to be ensured that these attributes exist in both the common view and application view of the implementor 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, for example, when receiving the reply. For instance, one of these returned In arguments can 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` on the Please Select In Arguments dialog and the Please Select Out Arguments dialog.

Exporting PL/SQL Code

To export PL/SQL code, use the File/Export menu item in iStudio.

The SQL Code dialog appears at the end of Subscribe, Invoke, Implement wizards. In the drop down list the options are generated data types and <EventType>_<EventName>_<MetadataOwner>_<Version>.

Typically, there is no need to modify the generated data types SQL code. Insert the necessary code as indicated by `-- declare here` or `-- fill code here` in the latter dialog.

Database Adapter Runtime Concepts

The following section describes the runtime concepts pertinent to the Database Adapter.

How the Database Adapter Works

The following topics describe how the Database adapter works.

Sending Adapter

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 Oracle9*i*AS 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 Oracle9*i*AS 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 appropriate 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. Therefore, if the adapter is down while the application is publishing Oracle9*i*AS InterConnect messages using the iStudio generated PL/SQL procedures, 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, the `DELETE FROM MESSAGEOBJECTTABLE` using SQLPlus can be run in the `oai` schema.

Receiving Adapter

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 appropriate PL/SQL procedures to inform the application about the newly arrived message. If this adapter were an implementor, the `OUT` arguments from the PL/SQL procedure invocation are put together and the REPLY in the form of another Oracle9*i*AS 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: *Oracle9iAS InterConnect User Guide*

Starting the Database Adapter

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

1. Access the Services window from the Start menu:

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

   The Services window displays.

2. Select the *OracleHome9iASInterConnectAdapter-Application* service.
3. Start the service based on your operating system:

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

See Also: "Configuring the Database Adapter" on page 2-6 for the location of the `start` script

Sample Log File

The following is a sample log file of a Database adapter that was successfully started.

```
D:\oracle\ora902\oai\9.0.2\adapters†bapp>D:\oracle\ora902\oai\9.0.2•in\JavaServi
ce.exe -debug "Oracle OAI Adapter 9.0.2 -
dbapp" D:\oracle\ora9021\oai\9.0.2\adapters†bapp adapter.ini
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.
```

Stopping the Database Adapter

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

1. Access the Services window from the Start menu:

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

   The Services window displays.

2. Select the *OracleHome9iASInterConnectAdapter-Application* service.
3. Start the service based on your operating system:

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

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

4

Sample Use Cases

This chapter describes sample use cases for the Database adapter. This chapter discusses the following topics:

- Database Adapter Sample Use Cases

Database Adapter Sample Use Cases

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 appropriate metadata in iStudio. This is usually `V1` unless the metadata versioning features was used in iStudio.

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 appropriate 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.

Design Time Steps

The following section describes metadata creation using iStudio.

> **See Also:** *Oracle9iAS InterConnect User Guide*

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

c. Click Add to add the following attributes:

* id (NUMBER)
* address (Address) [ARRAY]

4. Create an application in iStudio. Enter demopub in the Application Name field on the Create Application dialog.
5. Create a Published Event using the Publish Wizard in iStudio:
   a. Select demopub for the Application and Database as the Message Type on the Select an Event page.
   b. Expand the tree in the Select an Event and select createCustomer.
   c. Click Import on 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 on the Create Application dialog.
7. Create a Subscribed Event using the Subscribe Wizard in iStudio.
   a. Select demosub for the Application and Database as the Message Type on the Select an Event page.
   b. Expand the tree in the Select an Event box and select createCustomer.
   c. Click Import on 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. On 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 for the File Prefix.

      The following files are created and stored in the `$ORACLE_HOME/oai/4.1/iStudio` directory:

      * `demo_demopub_Customer.sql`
      * `demo_demopub_CustomerTYPES.sql`
      * `demo_demosub_Customer.sql`
      * `demo_demosub_CustomerTYPES.sql`

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 4-5

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/oai/9.0.2/bin` directory to create the demosub adapter. Usage: `copyAdapter demopub demosub`.

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;
/
```

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 appropriate 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.

Synchronous Invoke 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:** *Oracle9iAS InterConnect User Guide*

Design Time Steps

1. Create a business object in iStudio. On 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. On 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`.
      * On 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 on the Select a Procedure page.
   b. Expand the tree in the Select a Procedure box 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: Chapter 5, "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. On the Define Stored Procedure page, do not edit the SQL code; it is correct.
9. Click Finish.
10. Create an application in iStudio. On 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 tree in the Select a Procedure box and select newCustomer.

c. Click Import and select Database on 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`.
* On 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 on 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 on the Define Mapping OUT Arguments page:

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

14. Edit the SQL code on 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. Export SQL code by selecting Export from the File menu in iStudio. Select demoinv and demoimp from the context menu.

17. Enter demo for the File Prefix.

The following files are created and stored in the `$ORACLE_HOME/oai/4.1/iStudio` directory:

- `demo_demopub_Customer.sql`
- `demo_demopub_CustomerTYPES.sql`

- demo_demosub_Customer.sql
- demo_demosub_CustomerTYPES.sql

Runtime Steps The following 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.

See Also: "Related Files for Synchronous Invoke Implement" on page 4-13

Bring up two SQL prompts:

1. At the first prompt, connect as foo / manager.
2. Run the following SQL scripts:
   - @demo_demoinv_CustomerTYPES, @demo_demoinv_Customer
   - @demo_sync_invoke
3. At the second 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 have a trigger to update automatically when a new row is created in the correlation table.

Asynchronous Invoke 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: *Oracle9iAS InterConnect User Guide*

Design Time Steps

1. Create a business object in iStudio. Enter Customer in the business object name field On the Create Business Object dialog.
2. Create a common data type.
3. Create a procedure in iStudio. On 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.
      * On 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 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 on the Select a Procedure page.
   b. Expand the tree in the Select a Procedure box 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: Chapter 5, "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 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. 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>.id;`

9. Click Finish.

10. Create an application in iStudio. Enter demoimp in the Application Name field On 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 tree in the Select a Procedure box and select newCustomer.

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

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

    * Expand BAR, Tables / Views and select BAR.RESULTS.
    * On 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 on 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 on the Define Mapping OUT Arguments page:
    - `newCustomer:OUT [dempimp View] -- Object Copy -- newCustomer:OUT [Common View]`
14. Edit the SQL code on 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. Export SQL code by selecting Export from the File menu in iStudio. Select demoinv and demoimp from the context menu.
17. Enter demo for the File Prefix.

    The following files are created and stored in the `$ORACLE_HOME/oai/4.1/iStudio` directory:
    - `demo_demopub_Customer.sql`
    - `demo_demopub_CustomerTYPES.sql`
    - `demo_demosub_Customer.sql`
    - `demo_demosub_CustomerTYPES.sql`

Runtime Steps

Bring up two SQL prompts:

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

Related Files for Synchronous Invoke Implement

The following scripts are related to the runtime steps described in both cases in case two.

- `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);
```

Related Files for Asynchronous Invoke Implement

- 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);
```

- 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;
/
```

5

Frequently Asked Questions

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

- Installation Questions
- Design Time Questions
- Runtime Questions

Installation Questions

The following questions address installation of the Database adapter.

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 Oracle9*i*AS InterConnect.

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

Edit the `adapter.ini` file located in the `$ORACLE_HOME/oai/9.0.2/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 machine?

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/oai/9.0.2/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.

Design Time Questions

The following are design time questions for the Database adapter.

Where is the PL/SQL code exported through iStudio saved?

The PL/SQL code is save in the `$ORACLE_HOME/oai/4.1/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>_<BusinessObject>TYPES.sql
<PrefixSpecifiedInIStudio>_<ApplicationName>_<BusinessObject>.sql
```

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

Please see "Returned In Arguments" on page 3-5.

How do I export stored procedures to use with the Database adapter?

The following steps describe how to export procedures for the Database adapter.

1. Use iStudio and select Export from the File menu. The Export Stored Procedures dialog displays.
2. Select the root of the tree to select all stored procedures from all applications.

   Two files are created for each application and business object if a base name is selected. The `<base name>_<application name>_<business object name>.sql` file defines all stored procedures for all messages from the selected business object and application. The `<base name>_<applicationname>_<business object name>TYPES.sql` file defines all types used by the stored procedures in the first file.

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 tree.

See Also: "Runtime Questions" on page 5-4

Can database messages contain arrays of arrays?

The database does not allow arrays of arrays. Therefore, the application view of database messages should not contain arrays of arrays. For example, the application view of a database message can contain an array of Customers which each contained one Address. However, it can not contain an array of Customers which each contained an array of Addresses.

Runtime Questions

The following questions address runtime concepts for the Database adapter.

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

A start executable that is not the Oracle9*i*AS InterConnect `start` script must be running. This is dependent on what is in the PATH environment variable. Therefore, 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?

Make sure you none of the PL/SQL reserved keywords are used in Oracle9*i*AS 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.

See Also: "Design Time Questions" on page 5-2

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 username, 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.

See Also: "Design Time Questions" on page 5-2

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 if the Database adapter was started properly. This can be viewed in the `oailog.txt` file in the logs directory of the Database adapter.

Index

J

O

P

R

S

U