Oracle® Application Server InterConnect

Adapter for Oracle Applications Installation and User's Guide

10*g* (9.0.4.0.2)

Part No. B14114-01

June 2004

ORACLE®

Oracle Application Server InterConnect Adapter for Oracle Applications Installation and User's Guide, 10*g* (9.0.4.0.2)

Part No. B14114-01

Primary Author: Pradeep Vasudev, Vimmy K Raj

Contributor: Ashwin Patel, Rahul Pathak, Maneesh Joshi, Harish Sriramulu

Contents

Send Us Your Comments

Oracle Application Server InterConnect Adapter for Oracle Applications Installation and User's Guide, 10*g* (9.0.4.0.2)

Part No. B14114-01

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

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

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

- Electronic mail: appserverdocs_us@oracle.com
- FAX: 650.506.7375 Attn: Oracle Application Server Documentation Manager
- Postal service:

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

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

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

Preface

This Preface contains these topics:

Audience

Oracle Application Server InterConnect Adapter for Oracle Applications Installation and User's Guide is intended for system administrators of OracleAS InterConnect who perform the following tasks:

- install applications
- maintain applications

To use this document, you need to know installation and configuration of OracleAS InterConnect.

Documentation Accessibility

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

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Structure

This document contains:

Chapter 1, "Introduction"

This chapter describes the OracleAS InterConnect OA adapter, and its hardware and software requirements.

Chapter 2, "Installation and Configuration"

This chapter describes installation and configuration of the OA adapter.

Chapter 3, "Design Time and Runtime Concepts"

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

Chapter 4, "Sample Use Cases"

This chapter provides sample use cases for the OA adapter.

Chapter 5, "Frequently Asked Questions"

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

Related Documents

For more information, see these Oracle resources:

- *Oracle Application Server InterConnect User's Guide*
- *Oracle Application Server InterConnect Installation Guide*
- *Oracle Application Server InterConnect Adapter Configuration Editor User's Guide*

Printed documentation is available for sale in the Oracle Store at

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

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

`http://otn.oracle.com/membership/`

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/documentation/`

Conventions

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

Conventions in Text

We use the following 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.	*Oracle10g 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, Recovery Manager 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 executable programs, filenames, directory names, and sample user-supplied elements. *Note:* Some programmatic elements use a mixture of `UPPERCASE` and `lowercase`. Enter these elements as shown.	Enter `sqlplus` to start SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run *`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

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

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

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

Convention	Meaning	Example
[ ]	Anything enclosed in brackets is optional.	DECIMAL (*digits* [, *precision*])
{ }	Braces are used for grouping items.	{ENABLE | DISABLE}
|	A vertical bar represents a choice of two options.	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	Ellipsis points mean repetition in syntax descriptions.	CREATE TABLE ... AS *subquery*;
	In addition, ellipsis points can mean an omission in code examples or text.	SELECT *col1*, *col2*, ... , *coln* FROM employees;
Other symbols	You must use symbols other than brackets ([ ]), braces ({ }), vertical bars (|), and ellipsis points (...) exactly as shown.	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*
UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. Because these terms are not case sensitive, you can use them in either UPPERCASE or lowercase.	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
lowercase	Lowercase typeface indicates user-defined programmatic elements, such as names of tables, columns, or files. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;

Conventions for Windows Operating Systems

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

Convention	Meaning	Example
Choose **Start** > *menu item*	How to start a program.	To start the 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 filename begins with \\, then Windows assumes it uses the Universal Naming Convention.	c:\winnt"\"system32 is the same as C:\WINNT\SYSTEM32

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

1 Introduction

This chapter provides an overview on how to use Oracle Application Server InterConnect (OracleAS InterConnect) Adapter for Oracle Applications (OA adapter).

This chapter contains the following:

1.1 Oracle Applications Overview

Oracle Applications is a complete set of integrated business applications that runs entirely on the Internet. This enables you to:

- Cut costs
- Increase revenues across front-office and back-office functions
- Access current, accurate, and consistent data

The applications in Oracle Applications are built on a unified information architecture that consolidates data from Oracle and non-Oracle applications and allows a consistent definition of customers, suppliers, partners, and employees across the entire enterprise. The result is a suite of applications that can give you current performance metrics, financial ratios, profit and loss summaries, and other types of information that roll up across all departments, products, and geographies. Whether you implement one module or the entire Suite, Oracle Applications can help you make smarter decisions with better information, share unified information across the enterprise, reduce IT expenses, and run your business more efficiently.

To connect Oracle Applications to other non-Oracle applications, the OracleAS InterConnect OA adapter is required.

1.2 Oracle Applications Adapter Overview

The OracleAS InterConnect OA adapter enables you to connect to Oracle Applications. The OA adapter can be used to connect to versions 11.5.1 through 11.5.9 of Oracle Applications.

Note: All interface types except PL/SQL APIs and interface tables exposed by the Oracle Applications can be used for both inbound (into Oracle Applications) and outbound (out of Oracle Applications) communication. PL/SQL APIs and interface tables can be used only for inbound communication.

1.2.1 Hardware Requirements

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

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

1.2.2 Software Requirements

The following are software requirements for the OA adapter:

- Operating System Requirements
- JRE Requirements
- Database Requirements

1.2.2.1 Operating System Requirements

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

Table 1–1 Operating System Requirements

Operating System	Version
Windows NT	Version 4.0 with Service Pack 6 or above.
Windows 2000	With Service Pack 1 or above.
IBM AIX 5L	5.2, Maintenance Level 1 or higher
HP Tru64	UNIX V5.1B
HP-UX	11.0 and 11*i* (11.11) PA-RISC or higher
Linux	Red Hat Enterprise Linux AS/ES 2.1, AS/ES 3.0, UnitedLinux 1.0
Sun SPARC Solaris	8 and 9 (32 bit)

See Also: Refer to *OracleAS InterConnect Installation Guide* for more information

1.2.2.2 JRE Requirements

OracleAS InterConnect uses JRE 1.4.1, which is installed with its components.

1.2.2.3 Database Requirements

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

Note: There is no dependency between OracleAS InterConnect and the technology stack under Oracle Applications, specifically the Oracle Database and Application Server versions. You can use the OA adapter to connect to any of the above mentioned Oracle Applications versions.

2

Installation and Configuration

This chapter describes the installation and configuration of the OA adapter. This chapter contains the following topics:

- Installing the OA Adapter
- OA Adapter Configuration Parameters

2.1 Installing the OA Adapter

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

- An existing Oracle home for OracleAS InterConnect 10*g* (9.0.4.0.2).
- A new Oracle home. Oracle Universal Installer (OUI) creates this Oracle home.

This section contains the following topics:

- Preinstallation Tasks
- Installation Tasks
- Post Installation Steps

2.1.1 Preinstallation Tasks

Consult the following guides before installing the OA adapter:

- *Oracle10g Application Server Installation Guide* for information about OUI startup.
- *Oracle Application Server InterConnect Installation Guide*, for information about mounting CD-ROMs, listing software, hardware, and system requirements for OracleAS InterConnect, and installing OracleAS InterConnect.

> **Note:** OracleAS InterConnect hub is installable through the OracleAS InterConnect hub installation type. You must install the OracleAS InterConnect hub before proceeding with the OA adapter installation.

2.1.2 Installation Tasks

To install the OA adapter:

1. On the Available Product Components page of the OracleAS InterConnect installation, select **OA adapter**, and click **Next**.
   - Go to step 2, if installing the OA adapter in an independent Oracle home. Ensure that the OracleAS InterConnect hub has been installed.

- Go to step 3, if installing the OA adapter in an existing Oracle home. Ensure that it is a home directory to an OracleAS InterConnect component.

2. The OracleAS InterConnect Database configuration page is displayed. 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.
   * Password - The password for the hub database user.
3. Click **Next**. The Oracle Applications Adapter configuration page is displayed. Enter the application name. Blank spaces are not permitted. The default value is `myOAApp`.
4. Click **Next**. The Oracle Applications Database - Specify Database Connection Information page is displayed. This configures the information to the spoke application database. Enter information in the following fields:
   - Host Name - The name of the machine where the application database is installed.
   - Port Number - The TNS listener port for the application database.
   - Database SID - The SID for the application database.

   The information on this page is for Oracle Applications, from which the adapter will deliver or receive messages.
5. Click **Next**. The Oracle Applications Database - Specify APPS Schema Password page is displayed. Enter the password for the APPS schema name.
6. Click **Next**. The Summary page is displayed.
7. Click **Install** to install the OA adapter and other selected components. The OA adapter is installed in the following directory:

Platform	Directory
Windows	*ORACLE_HOME*\oai\9.0.4\adapters*Application*
UNIX	*ORACLE_HOME*/oai/9.0.4/adapters/*Application*

   Application is the value specified in Step 3.
8. Click **Exit** on the Installation page to exit the OA adapter installation.

2.1.3 Post Installation Steps

After installation, a set of post-installation steps is displayed. These steps are also copied to the `post-installation.txt` file in the OA adapter directory.

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

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

1. Access the directory where the OA adapter is installed.

2. Execute the following for the correct database version:

Database Version	Code to Execute
Oracle8*i*	`oaischema -create -8i sys/[password] [tnsname]`
Oracle9*i*	`oaischema -create sys/[password] [tnsname]`

where:

- password—The password for the system user.
- tnsname—The tnsname for the Oracle OA instance.

2.2 OA Adapter Configuration Parameters

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

Table 2–1 OA Adapter Directory

Platform	Directory
UNIX	*ORACLE_HOME*`/oai/9.0.4/adapters/Application`
Windows	*ORACLE_HOME*`\oai\9.0.4\adapters\Application`

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

Table 2–2 OA Executable Files

File	Description
`start.bat` (Windows) `start` (UNIX)	Takes no parameters, starts the adapter
`stop.bat` (Windows) `stop` (UNIX)	Takes no parameters, stops the adapter
`ignoreerrors.bat` (Windows) `ignoreErrors` (UNIX)	If an argument is specified, then the given error code is ignored. For example: `ignoreerrors` *`errorCodeToBeIgnored`* If no argument is specified, then all error codes specified in the `ErrorCodes.ini` file are ignored. For example: `ignoreerrors`

Table 2–3 describes the OA adapter configuration files.

Table 2–3 OA Configuration Files

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

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

Table 2–4 OA Directories

Directory	Description
`persistence`	The messages are persisted (made available) in this directory. Do not edit this directory or its files.
`logs`	The adapter activity is logged in subdirectories of the `logs` directory. Subdirectory names take the following form: *`timestamp_in_milliseconds`* Each time the adapter is run, a new subdirectory is created in which logging is done in an `oailog.txt` file.

2.2.1 Hub.ini Parameter File

The OA adapter connects to the hub database using the parameters from the `hub.ini` file. Table 2–5 lists the parameter name, description, and an example for each parameter.

Table 2–5 Hub.ini Parameters

Parameter	Description	Example
hub_username	The name of the hub database schema (or username). The default value is `oaihub904`.	`hub_username=oaihub904`
hub_password	The password for the hub database user. There is no default value. The value is set during installation.	`hub_password=manager`
hub_host	The name of the machine 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`
repository_name	The name of the repository that communicates with the adapter. The default value is `InterConnectRepository`.	`repository_name=InterConnectRepository`

2.2.1.1 Oracle Real Application Clusters Hub.ini Parameters

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

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

Parameter	Description	Example
hub_num_nodes	Number of nodes in Real Application Clusters.	`hub_num_nodes=4`
hub_hostx	The host where the Real Application Clusters database is installed.	`hub_host2=dsunram13`
hub_instancex	The instance on the respective node.	`hub_instance2=orcl2`
hub_portx	The port on which the TNS listener is listening.	`hub_port2=1521`

2.2.2 Agent Connection Parameters

The agent component of the OA adapter reads the `adapter.ini` file at runtime to access information on configuring the OA adapter parameter. Table 2–7 lists the parameter name, the description, and an example for each parameter.

Table 2–7 Agent Connection Parameters

Parameter	Description	Example
application	Specifies the name of the application to which the adapter connects. This must match with the name specified in iStudio while creating metadata. Use any alphanumeric string. There is no default value.	`application=Ebizapp`
partition	Specifies the partition the adapter handles as specified in iStudio. Use any alphanumeric string. There is no default value.	`partition=germany`
instance_number	Specifies the instance number to which the adapter corresponds. Specify a value only if you want to have multiple adapter instances for the given application with the given partition. Use any integer greater than or equal to `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	Specifies the subscriber name used when the adapter registers its subscription. The possible value is a valid Oracle Advanced Queue subscriber name. There is no default value.	`agent_subscriber_name=Ebizapp`
agent_message_selector	Specifies conditions for message selection when the adapter registers its subscription with the hub. The possible value is a valid Oracle Advanced Queue message selector string (like `'%,aqapp,%'`). There is no default value.	`agent_message_selector=%,Ebizapp,%`
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. The possible value is the application name (*`parameter`*:*`application`*) concatenated with the instance number (*`parameter`*:*`instance_number`*). There is no default value.	If `application=Ebizapp` and `instance_number=2`, then `agent_reply_subscriber_name=Ebizapp2`
agent_reply_message_selector	Specifies the application instance to which the reply must be sent. Used only if multiple adapter instances exist for the given application and given partition. The value is a string built using the application name (*`parameter`*:*`application`*) concatenated with the instance number (*`parameter`*:*`instance_number`*). There is no default value.	If `application=Ebizapp` and `instance_number=2`, then `agent_reply_message_selector=recipient_list`, like `'%,Ebizapp2,%'`
agent_tracking_enabled	Specifies if message tracking is enabled. Set this parameter to `false` to turn off all tracking of messages. Set this parameter to `true` to track messages with tracking fields set in iStudio. The default value is `true`.	`agent_tracking_enabled=true`

Table 2–7 (Cont.) Agent Connection Parameters

Parameter	Description	Example
agent_throughput_measurement_enabled	Specifies if the throughput measurement is enabled. Set this parameter to `false` to turn off all throughput measurements. The default value is `true`.	`agent_throughput_measurement_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 InterConnect DTD for all messages sent to the hub. Set this parameter to `true` if you want the adapter to use the DTD imported for the common view messages instead of the OracleAS InterConnect DTD. The values are `true` or `false`. There is no default value.	`agent_use_custom_hub_dtd=false`
agent_metadata_caching	Specifies the metadata caching algorithm. The values are: ■ `startup` - Cache everything at startup. This may take a while if there is metadata 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`
agent_dvm_table_caching	Specifies the Domain Value Mapping (DVM) table caching algorithm. Possible values are: ■ `startup` - Cache all DVM tables at startup. This may take a while if there are many 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. The values are: ■ `startup` - Cache all lookup tables at startup. This may take a while if there are many 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	Specifies whether the agent should delete the file cache when it starts. 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. The values are `true` or `false`. The default value is `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_max_ao_cache_size	Specifies the maximum number of application object metadata to cache. The value is any integer greater than or equal to `1`. The default value is `200`.	`agent_max_ao_cache_size=200`
agent_max_co_cache_size	Specifies the maximum number of common object metadata to cache. The value is any integer greater than or equal to `1`. The default value is `100`.	`agent_max_co_cache_size=100`

Table 2–7 (Cont.) Agent Connection Parameters

Parameter	Description	Example
agent_max_message_metadata_cache_size	Specifies the maximum number of message metadata (publish/subscribe and invoke/implement) to cache. The value is any integer greater than or equal to 1. The default value is 200.	agent_max_message_metadata_cache_size=200
agent_max_dvm_table_cache_size	Specifies the maximum number of DVM tables to cache. Possible value is any integer greater than or equal to 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. The values is any integer greater than or equal to 1. The default value is 200.	agent_max_lookup_table_cache_size=200
agent_max_queue_size	Specifies the maximum size to which internal OracleAS InterConnect message queues can grow. The value is any integer greater than or equal to 1. The default value is 1000.	agent_max_queue_size=1000
agent_persistence_queue_size	Specifies the maximum size of internal OracleAS InterConnect persistence queues. The value is any integer greater than or equal to 1. The default value is 1000.	agent_persistence_queue_size=1000
agent_persistence_cleanup_interval	Specifies how often to run the persistence cleaner thread in milliseconds. The value is any integer greater than or equal to 30000 milliseconds. The default value is 60000.	agent_persistence_cleanup_interval=60000
agent_persistence_retry_interval	Specifies how often the persistence thread retries when it fails to send an OracleAS InterConnect message. The value is any integer greater than or equal to 5000 milliseconds. The default value is 60000.	agent_persistence_retry_interval=60000
agent_pipeline_to_hub	Specifies whether to turn on or off the pipeline for messages from the bridge to the hub. If you set the pipeline to false, the the file persistence is not used in that direction.	agent_pipeline_to_hub=false
agent_pipeline_from_hub	Specifies whether to turn on or off 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.	agent_pipeline_from_hub=false
service_path	Specifies the value to which the environment variable PATH is to be set. The PATH variable is set to the specified value before starting the Java VM. Typically, list all directories here that contain necessary DLLs. Possible value is the valid PATH environment variable setting. There is no default value. This parameter is for Windows only.	service_path=%JREHOME%\bin;D:\oracle\ora904\bin
service_classpath	Specifies the class path used by the adapter Java VM. If a custom adapter is developed and, as a result, the adapter is to pick up any additional jars, add the jars to the existing set of jars being picked up. Possible values are the valid class path. There is no default value. This parameter is for Windows only.	service_classpath=D:\oracle\ora904\oai\904\lib\oai.jar;D:\oracle\ora904\jdbc\classes12.zip
service_class	Specifies the entry class for the Windows service. A possible value is oracle/oai/agent/service/AgentService. There is no default value. This parameter is for Windows only.	service_class=oracle/oai/agent/service/AgentService

Table 2–7 (Cont.) Agent Connection Parameters

Parameter	Description	Example
service_max_java_stack_size	Specifies the maximum size that the Java VM's stack can grow. The possible value is the valid Java VM maximum native stack size. The default value is same as the default value for the Java VM. This parameter is for Windows only.	`service_max_java_stack_size=409600`
service_max_native_stack_size	Specifies the maximum size that the Java VM's native stack can grow. The possible value is the valid Java VM maximum native stack size. The default value is same as the default value for the Java VM. This parameter is for Windows only.	`service_max_native_stack_size=131072`
service_min_heap_size	Specifies the minimum heap size for the adapter Java VM. The possible value is any valid Java VM heap sizes. The default value is 536870912. This parameter is for Windows only.	`service_min_heap_size=536870912`
service_max_heap_size	Specifies the maximum heap size for the adapter Java VM. The possible value is any valid Java VM heap sizes. The default value is 536870912. This parameter is for Windows only.	`service_max_heap_size=536870912`
service_num_vm_args	Specifies the number of `service_vm_arg`*`number`* parameters specified in Java VM. The possible value is the number of `service_vm_arg`*`number`* parameters. There is no default value. This parameter is for Windows only.	`service_num_vm_args=1`
service_vm_argnumber	Specifies any additional arguments to the Java VM. For example, to retrieve line numbers in any of the stack traces, set `service_vm_arg1=java.compiler=NONE`. If a list of arguments exists, use multiple parameters as shown in the example, by incrementing the last digit starting with `1`. Set the `service_num_vm_args` parameter correctly. Possible value is any valid Java VM arguments. There is no default value. This parameter is for Windows only.	`service_vm_arg1=java.compiler=NONE` `service_vm_arg2=oai.adapter=database`
encoding	Specifies the character encoding for published messages. The adapter uses this parameter to generate encoding information in the encoding tag of transformed OracleAS InterConnect messages. OracleAS InterConnect represents messages internally as an XML document. The default encoding of the XML document is `UTF-8`. However, this encoding can be configured using this parameter, which is typically used when the OracleAS InterConnect message consists of characters not supported by `UTF-8` and when the `XMLParser` is unable to handle them.	`encoding=JA16SJIS`

Table 2–7 (Cont.) Agent Connection Parameters

Parameter	Description	Example

nls_date_format	Specifies the format for date fields expressed as string. The default date format is `EEE MMM dd HH:mm:ss zzz yyyy`. For the meaning of this string, refer to the list of reserved characters in Table 2–8.	Date format pattern `DD/MM/YYYY` can represent 01/01/2003.
		`nls_date_format=DD-MM-YYYY`
		Multiple date formats can be specified as `num_nls_formats=2`
		`nls_date_format1=dd-MM-yy`
		`nls_date_format2=dd/MM/yy`

nls_country	This parameter is a valid ISO country code. These codes are defined by ISO-3166. A full list of the codes is available at a Web site, such as, `http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html` The default country code is `US`. **Note**: This parameter specifies the date format. It is applicable for the date format only.	`nls_country=US`
nls_language	This parameter is a valid ISO language code. These codes are defined by ISO-639. A full list of these codes is available at a Web site, such as, `http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt` The default language code is `en`. **Note**: This parameter specifies the date format. It is applicable for the date format only.	`nls_language=en`

Table 2–8 shows the reserved characters used to specify the value of the `nls_date_format` parameter. Using these characters, you can construct a pattern to define date formats.

Table 2–8 Reserved Characters for the nls_date_format Parameter

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

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

Letter	Description	Example
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`

2.2.2.1 OA Adapter-specific Parameters

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

Table 2–9 OA Adapter-specific Parameters

Parameter	Description	Example
bridge_class	Specifies the entry class for the OA adapter. The value cannot be modified later. The default value is `oracle.oai.agent.adapter.ebs.EBSBridge`.	`bridge_ class=oracle.oai.agen t.adapter.ebs. EBSBridge`
ebs_bridge_use_thin_ jdbc	Specifies whether to use a thin JDBC driver when talking to the APPS database. The values are true and false. The default is `true`.	`EBS_bridge_thin_ jdbc=true`
ebs_bridge_sql_ trace	Specifies whether to turn sql tracing on for the APPS database. The values are true or false; the default value is `false`.	`ebs_bridge_sql_trace =true`
ebs_bridge_schema_ username	Specifies the username for the schema number `<schema#>`. The values for the schema number are 1 through `<ebs_bridge_num_schemas>`. This value should not be modified. Possible values are any valid database user name and there is no default value.	`ebs_bridge_schema1_ username=oai`
ebs_bridge_schema_ password	Specifies the password for the user in the `ebs_ bridge_schema<schema#>_username`. The value is the password for the corresponding database user. There is no default value.	`ebs_bridge_schema1_ password=oai` `encrypted_ebs_bridge_ schema1_ password=112511011064 109110871093`
ebs_bridge_schema_ host	Specifies the name of the machine hosting the database instance. The value is the name of the machine hosting the database. There is no default value.	`ebs_bridge_schema_ host=dlsun4255`
ebs_bridge_schema_ port	Specifies the port where the TNS listener is running for the database instance. The possible value is any valid TNS listener port number. There is no default value.	`ebs_bridge_schema_ port=1521`
ebs_bridge_schema_ instance	Specifies the SID of the database instance. The value is any valid SID. There is no default value.	`ebs_bridge_schema_ instance=oaimain`
ebs_bridge_schema_ num_readers	Specifies 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 values are any integer greater than 0. There is no default value.	`ebs_bridge_schema_ num_readers=1`

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

Parameter	Description	Example
ebs_bridge_schema_ num_writers	Specifies 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. The values are any integer greater than 0. There is no default value.	`ebs_bridge_schema_ num_writers=1`
ebs_bridge_schema_ writer_username	Specifies the username to be used by this writer to log on to the database. There is no default value.	`ebs_bridge_schema_ writer_username=apps`
ebs_bridge_schema_ writer_password	Specifies the password corresponding to the database. There is no default value.	`ebs_bridge_schema_ writer_password=apps`
ebs_bridge_schema_ writer_use_oracle_ objects	Specifies whether to use Oracle Objects, available in Oracle8*i* and later releases.The values are true or false. The default value is `false`.	`ebs_bridge_schema1_ writer_use_oracle_ objects=true`
ebs_aq_bridge_owner	Specifies application (spoke) database queue (applicable for XML Gateway, WF Queue, and Custom Queue only).	`ebs_aq_bridge_ owner=apps`
ebs_aq_bridge_ consumer_name	If all the queues that this adapter will connect to on the application database side are single consumer queues, this can be left blank. If, however, any of the queues is a multiconsumer queue, then specify a consumer name. The possible value is the same as the ebs_bridge_ username. There is no default value.	`ebs_aq_bridge_ consumer_name=`ebsuser

3

Design Time and Runtime Concepts

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

3.1 OA Adapter Design Time Concepts

The OA adapter acts as a highly flexible integration interface to the Oracle Applications. This section describes the design time aspects of the OA adapter. The OA adapter provides the following interfaces to integrate with the Oracle Applications:

- Tables/Views/APIs

  This interface type should be used to integrate with Oracle Applications through direct database access using tables/views/APIs/ADTs. This interface type allows you to define your Application or Common Views using the database tables/views/APIs/ADTs in Oracle Applications. This interface shields integration code from underlying changes, which may exist between apps versions. As a result, it is suitable for performing DML operations against the Oracle Applications. Custom staging tables may also be used for inbound data to provide reporting and/or editing prior to application to base tables. Custom Object or XML views can be created which provide the structures necessary for producing outbound messages.

  This method is mainly used when the volume of data is large. Once the data is received in the table or by the API, you have to ensure that all messages (successful and errored) are backtracked.

- XML Gateway

  This interface type should be used to integrate with Oracle Applications using XML Gateway. This interface type allows you to define your Applciation or Common Views by importing one of the Oracle Applications prescribed DTDs. While publishing/subscribing events and invoking/implementing procedures, iStudio will automatically retrieve the standard queue header fields required for XML Gateway queue and define the Payload field (ANY type) with the imported DTD. Oracle provides out of the box implementations of OAG documents, which

can be integrated with AS InterConnect. The BOD's provided are tightly integrated with seeded workflows and API's. Thus, they can be quickly implemented with a minimum of custom development.

- Workflow Business Event System (BES)

  This interface type can be used to integrate with Oracle Applications using Workflow BES queue. This interface type allows you to define you Application or Common Views either importing database tables/views/APIs/ADTs (from Oracle Applications) or by importing a DTD file. While publishing/subscribing events and invoking/implementing procedures, iStudio will automatically retrieve the standard queue header fields required for Workflow BES queue and define the EVENT_DATA field (ANY Type) with the imported definition. BES in conjunction with Worfklow allows you to include other systems in seeded or custom workflows. Workflow is tightly integrated with many apps modules and is ideal for producing and consuming event based messages. BES/Worfklow may also be used to model long running and conversational type integration processes. Availability of predefined business events in Oracle Applications plays a big role in deciding which message can be logically grouped together instead of at the table level.

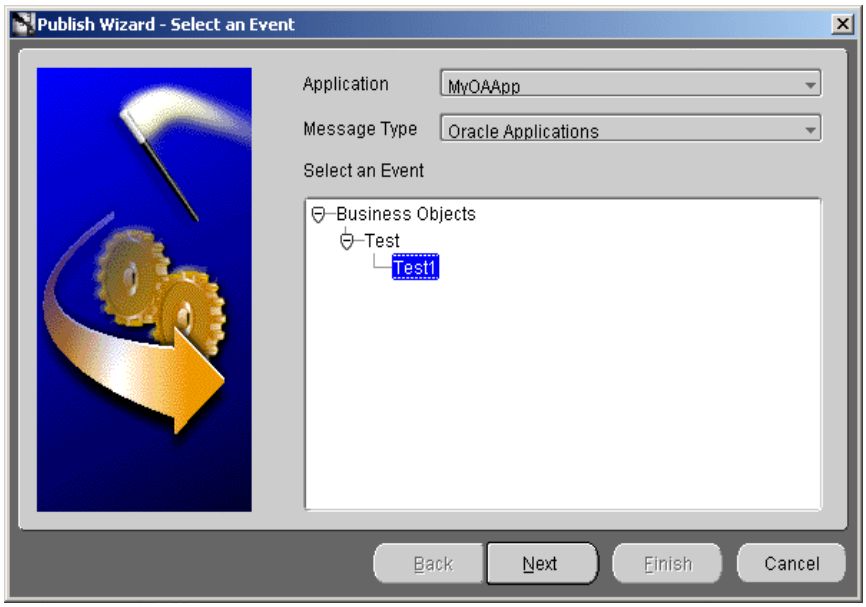
- Custom Queues

  This interface type can be used to integrate with the Oracle Applications using custom queues, if you are unable to use either XML Gateway or Workflow BES queue. This interface type allows you to define your Application or Common Views by either importing database tables/views/APIs/ADTs (from Oracle Applications) or by importing a DTD file. In this case, you are responsible for defining the queue headers, if required, as well as the payload part of the queue. Custom queues are often defined for very specific tasks or situations where very high throughput is required. User defined payloads can provide quick access to message metadata without the overhead of xml parsing. Custom queues may also be integrated with BES by developing queue handlers. This method is suitable when message payload is large and dynamic in nature. It also fits well with XML handling capabilities of the database and can be easily used with the Workflow both standalone and embedded.

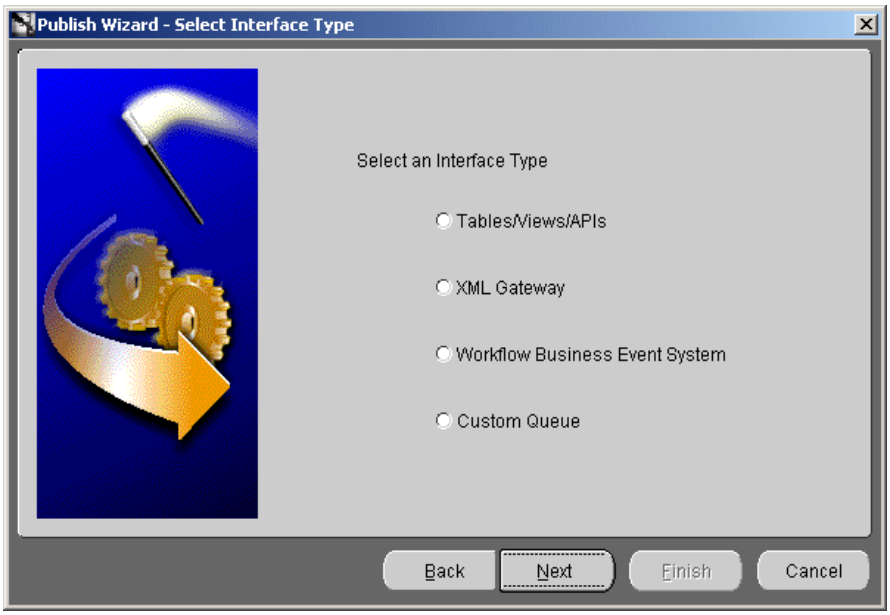
3.2 Designing with iStudio

OA adapter supports both the publish/subscribe and request/reply messaging paradigms. To create the message definitions, you must invoke the appropriate wizards. For example, to create a publish event, you must use the Publish wizard, as described in the following steps:

1. In the Design Navigation tree, expand the Application node. Select and expand the Application node to display the Published Events leaf. Right-click Published Events and select **New**. The Publish Wizard is displayed.



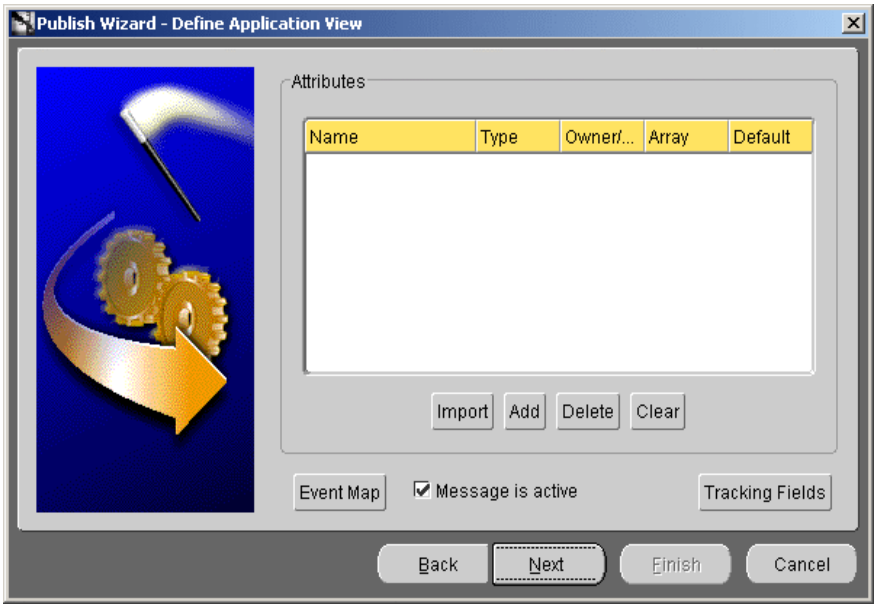
2. Enter following information in the fields:
   - **Application** - the name of the invoking application is selected by default.
   - **Message Type** - The mode of communication between OracleAS InterConnect and the application. Select the Oracle Applications message type.
3. Select the event name.
4. Click **Next**. The Select Interface Type screen is displayed.
5. Select the type of interface from which the definition is to be imported. The interface types are:



   - **Tables/Views/APIs**
   - **XML Gateway**

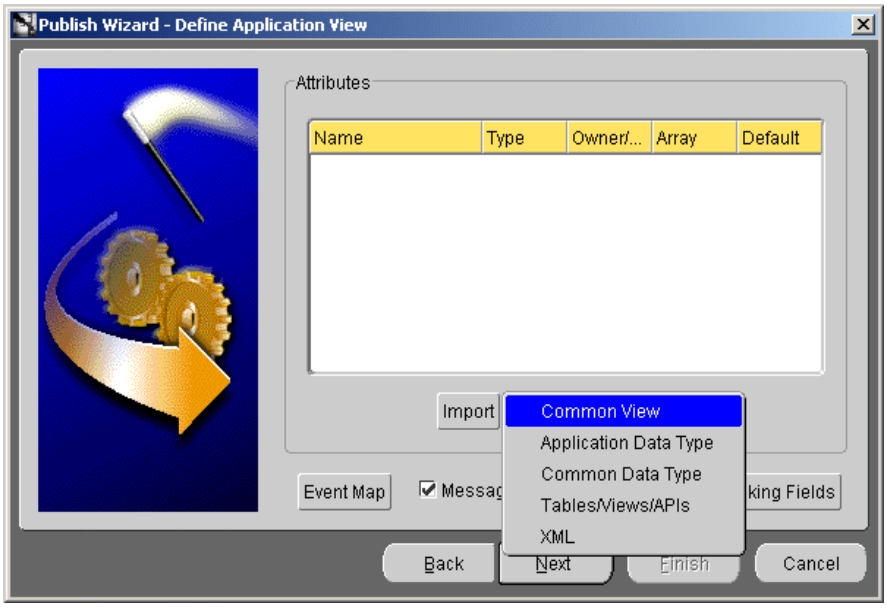
- **Workflow Business Event System**
- **Custom Queue**

6. Click **Next**. The Define Application View page is displayed.



Once an event is selected to publish, the application view is defined. The application view page is initially an empty table.

7. Click **Import** to import the definitions. A list of Import options are displayed.



Depending on the option chosen in step 5, the import menu options vary, as displayed in Table 3–1.

Table 3–1 Import Menu Options

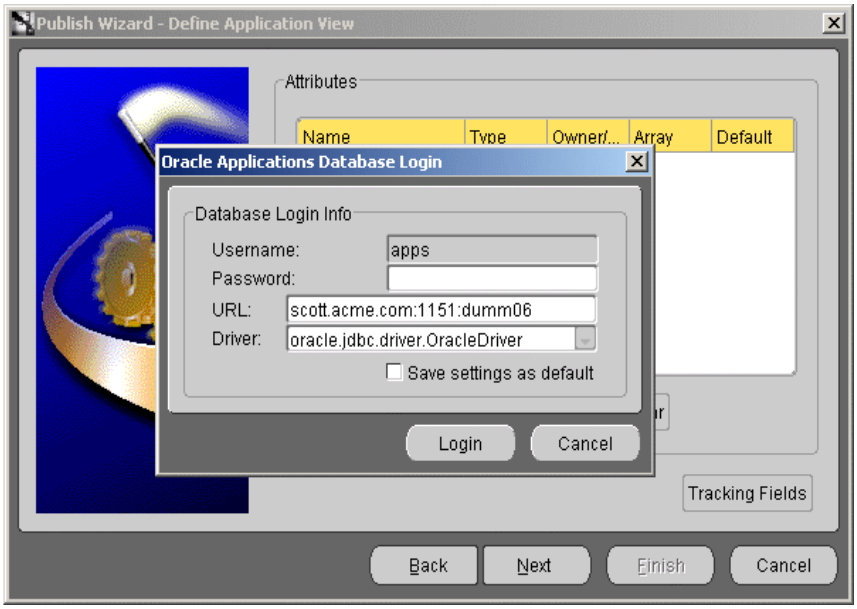
If you had chosen...	Import Menu Options are...
Tables/Views/APIs	Common View, Application Data Type, Common Data Type, Tables/Views/APIs
XML Gateway	Application Data Type, Common Data Type, XML Gateway, XML
Workflow Business Event System	Application Data Type, Common Data Type, Tables/Views/APIs, XML
Custom Queue	Common View, Application Data Type, Common Data Type, Tables/Views/APIs, XML

If you choose Common View, Application Data Type, or Common Data Type, the steps to be followed are identical to those explained in the *OracleAS InterConnect User's Guide*.

If you choose the XML option from the Import Menu, go to Step 10.

If you choose Tables/Views/APIs, the Oracle Applications Login screen is displayed as shown in Step 8.

8. Enter information in the following fields:



User Name—The database log in name. By default, the username for the application schema of the Oracle Applications is **apps**.

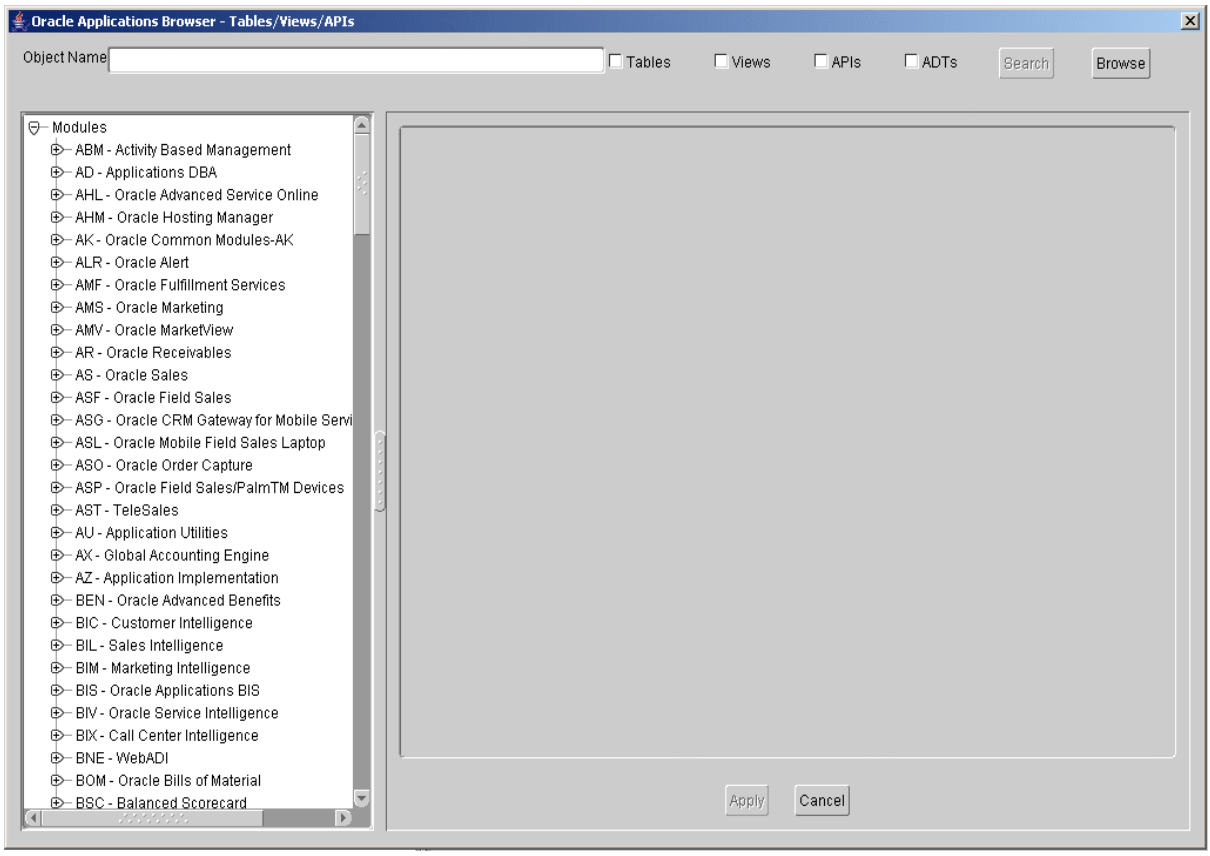
Password—The database log in password.

URL—The machine name: port number: database `SID`.

Driver—The JDBC driver used to connect to the database.

Save settings as default—Select this check box to save the settings for the workspace.

9. Click **Login**. The Oracle Applications Browser is displayed.



The browser displays the list of installed modules. You can use the following approaches to find the desired object:

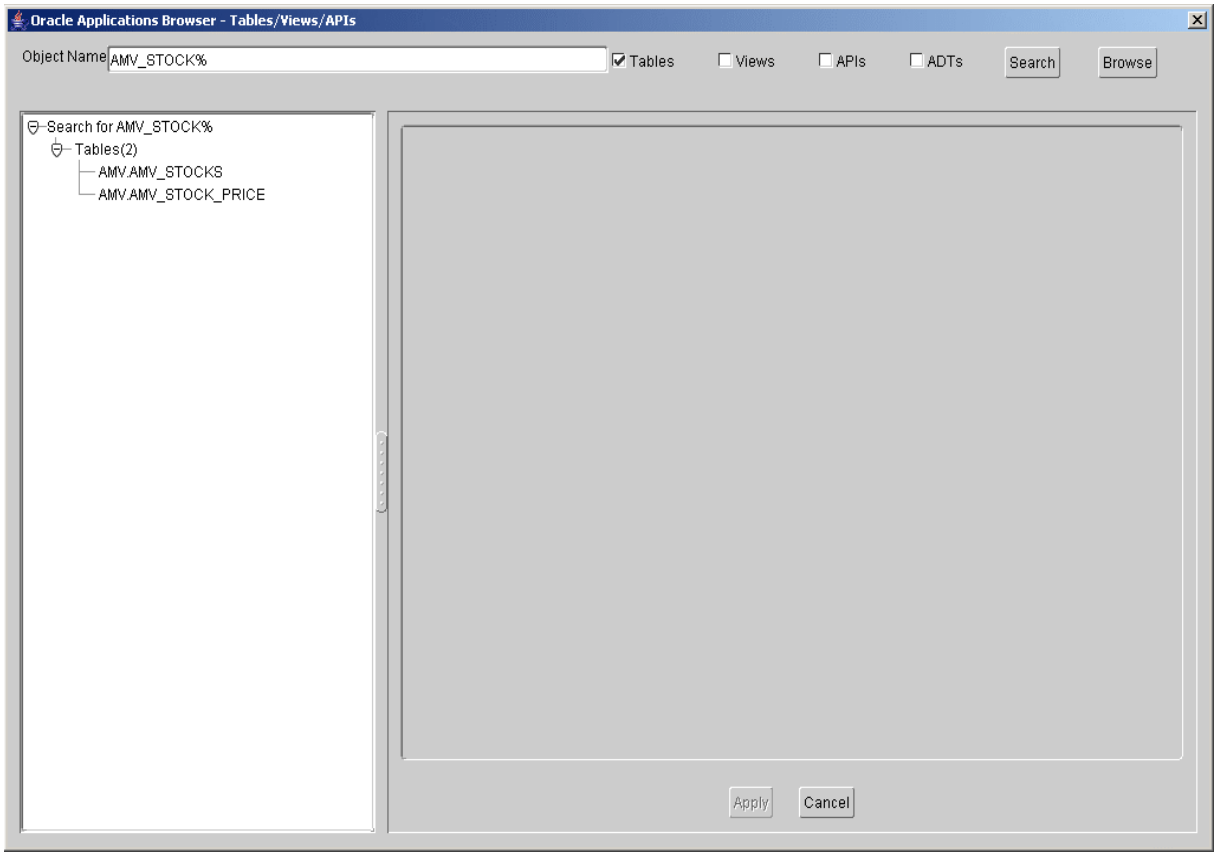
- **Browse Modules** — Browse through the list of modules displayed, and select the desired object. The attributes of the selected object are displayed on the right pane of the Browser. Select the desired attributes, and click **Apply**. The Define Application View dialog is displayed with the selected attributes. Go to Step 13 to continue.

  Note: For PL/SQL APIs, the browse mode only displays the PL/SQL packages that start with the module short name.

  For example, if the Oracle Receivables module has the short name "AR", only AR_<package name> packages will be displayed under this module. If the Oracle Receivables module has RA_<package name> and HZ_<package name> packages also, these will not be displayed under this module.

  For all other PL/SQL packages that logically belong to a module but do not show up where expected, use the search functionality.

- **Search Object** — Use the search facility to search for the desired object in the Oracle Applications schema. Enter the object name in the Object Name field, and select the appropriate object types (Tables, Views, APIs, ADTs).

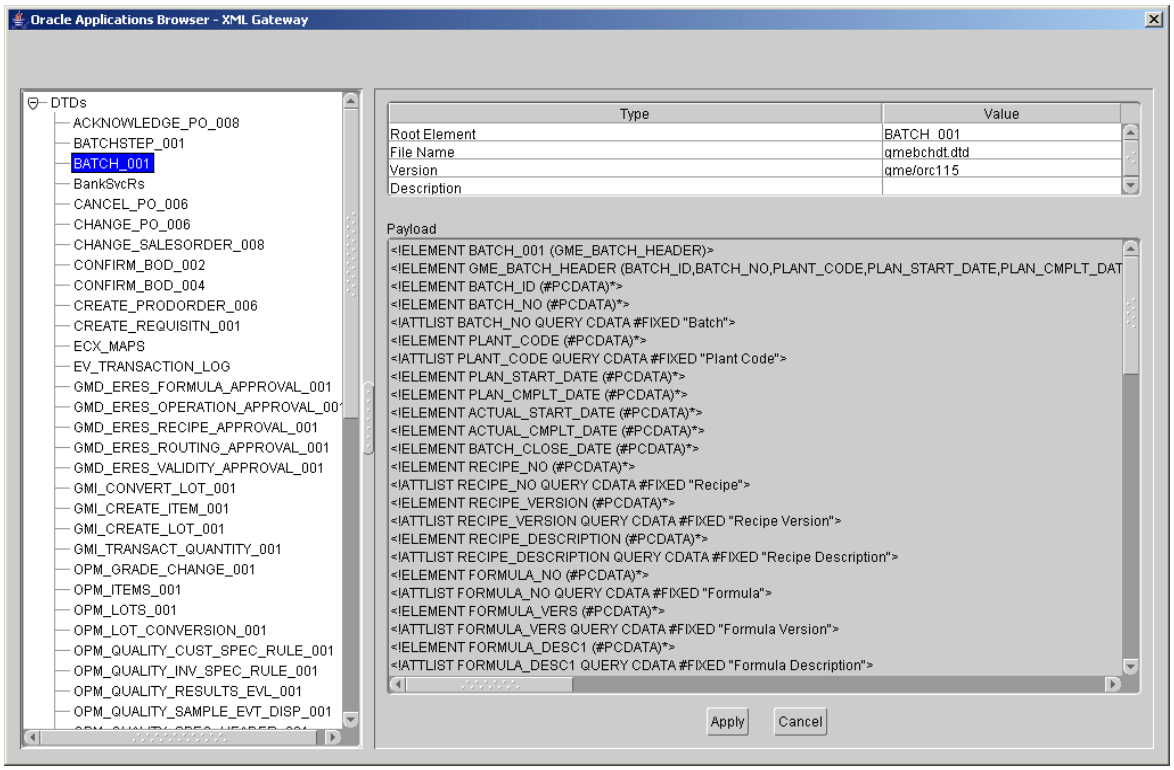


If you are not sure of the complete name of the desired object, you can enter the entire object name that you are looking for or use the wild card '%', for example, AMV_STOCK%. You must select at least one of the database object types listed at the top of the screen to enable the Search button.

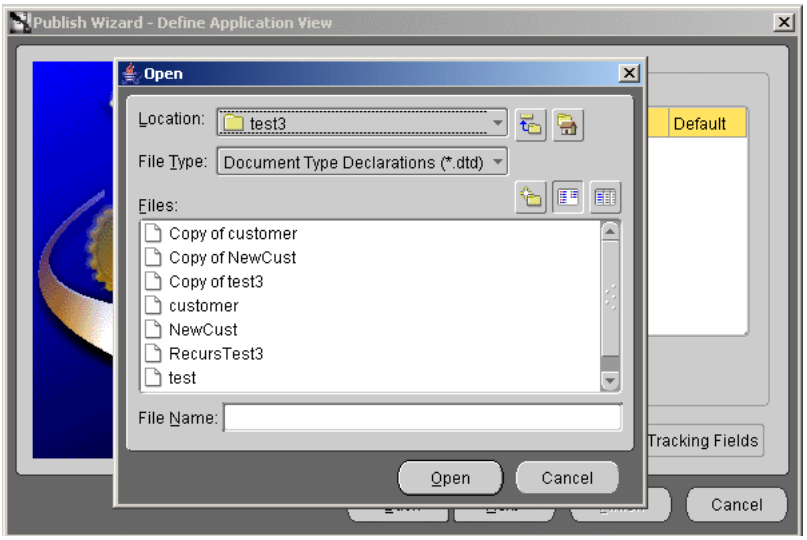
Click **Search** to display the list of matching object names. Select the desired object, the attributes of the selected object are displayed on the right pane of the Browser. Select the desired attributes, and click **Apply**. The Define Application View dialog is displayed with the selected attributes. Go to Step 13 to continue.

Note: To return from the Search option to the Browse option, click the **Browse** button.

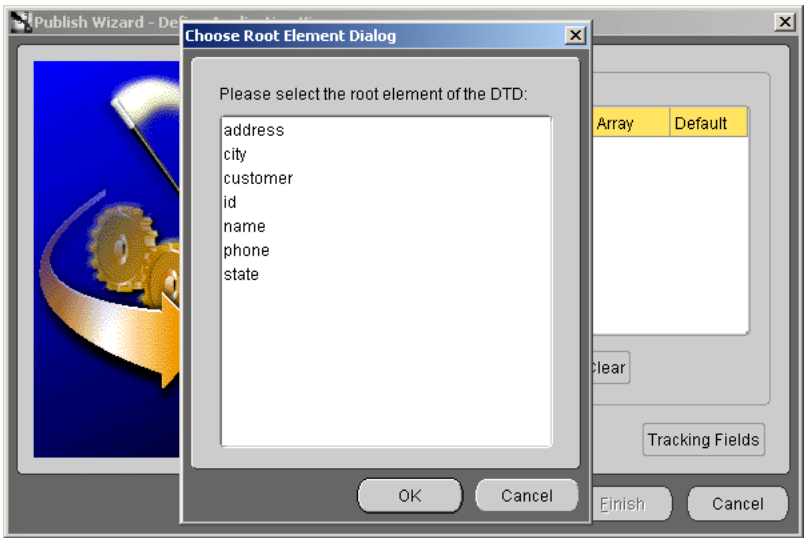
If you have chosen XML Gateway in Step 5, then the XML Gateway Browser is displayed. In the Browser, the left pane displays a list of DTDs and the right pane displays the details of the DTD such as its root element, file name, version, description, and payload. Also, the payload details of the selected DTD are displayed, as shown in the following figure. Select a DTD, and click **Apply**. The Define Application View dialog is displayed with the selected attributes. Go to Step 13 to continue.



10. The Open dialog is displayed. Select a DTD file, and click **Open**.

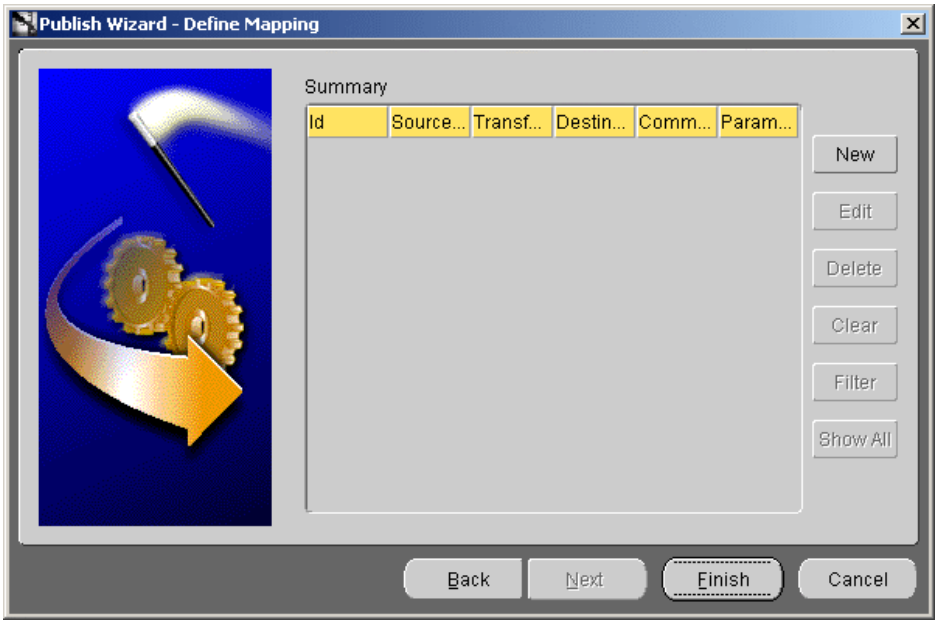


11. The Choose Root Element dialog is displayed. Select a root DTD element.

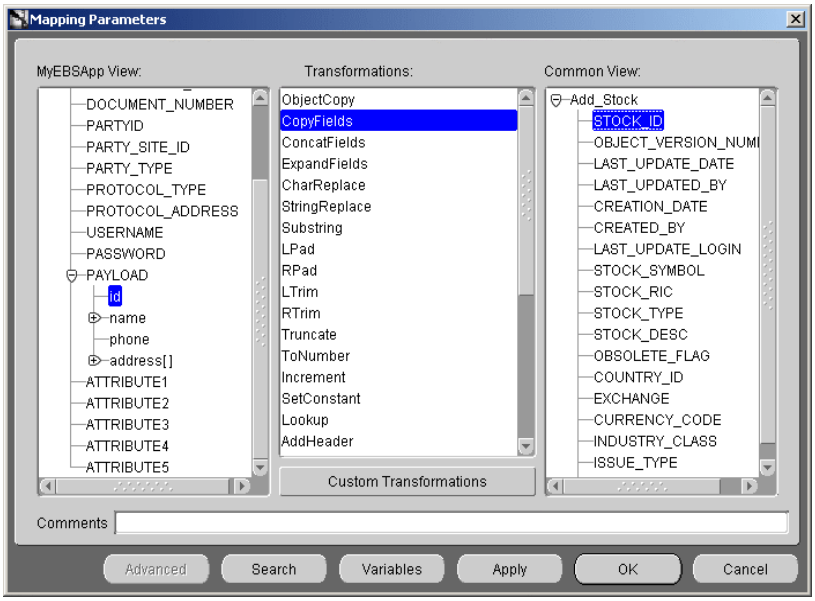


12. Click **OK**. The Define Application View dialog is displayed with the imported DTD.

13. Click **Next** in the Define Application View page. The Define Mapping page is displayed. Mapping involves copying the individual fields or simple shape-change transformations.



14. Click **New** to define new mappings. The Mapping Parameters dialog is displayed



Use a transformation to map fields in the application view to fields in the common view. For example, to map fields in the `FirstName` and `LastName` in the common view to `Name` in the application view, use the `ExpandFields` transform.

15. Click **OK** to return to the Publish Event Wizard.
16. Click **Finish**.

Once this is done, the Publish wizard exits. You have successfully created a publish event using iStudio, leveraging the OA adapter. The steps for the creation of subscribe events, as well as for invoking and implementing procedures, are similar to those described in the preceding steps.

3.3 OA Adapter Runtime Concepts

This section describes the key runtime components of the Oracle E-Business Suite adapter. You can choose one or more of the interface types to connect to a single Oracle Applications instance. The adapter is designed so that a single adapter instance can handle all the different interface types (inbound and outbound) for one application instance. You do not need to have dedicated adapters per interface type.

3.3.1 How the OA Adapter Works

This section contains these topics:

- OA Receiver
- OA Sender

3.3.1.1 OA Receiver

On the subscribing/receiving side, the OA adapter receives the message from the hub, transforms it from common view to application view, and passes it to the bridge that

- calls the appropriate PL/SQL procedures to inform the application about the newly arrived message if the interface type is Tables/Views/APIs.
- enqueues the message to the subscribe queue configured through the Deploy tab of iStudio if the message is for one of the queue interface types (XML Gateway,

Custom Queue, and Workflow BES). The application should then pick this message from this queue.

3.3.1.2 OA Sender

The OA adapter is comprised of the bridge and the runtime agent. This bridge has two different modes of communication with the Oracle Applications instance:

- Database Access: Interface Type - Tables/Views/APIs

  One part of the bridge component is constantly polling the `MESSAGEOBJECTTABLE` table in the oai schema, specified by the `oa_bridge_schema_username` parameter. A new row in this table indicates a new outbound OracleAS 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 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 OA adapter communicate in the form of direct database access 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 OracleAS 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 OA 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.

- Message Queues

  The second part of the OA adapter is constantly polling one or more queue(s) (XML Gateway, Workflow BES, Custom Queue) chosen for publishing messages in the `apps` schema. A new message in this queue indicates a new outbound OracleAS InterConnect message waiting to be sent by the adapter. The adapter then picks up the message, builds the corresponding OracleAS InterConnect message, persists it, transforms it to the common view, and routes it to the hub. From the hub, the message is routed to the appropriate subscriber based on configuration done using iStudio, which could be content-based, or subscription-based.

  The application and the OA adapter communicate via the publishing and invoking queues, residing in the `apps` schema for outbound messages and via subscribing and implementing queues for inbound messages. Therefore, the OA adapter is down while the application is publishing OracleAS InterConnect messages. These messages are held in the queues and will be picked up in the order they were enqueued by the OA adapter once it is up and running. If there are messages in the queues, which should no longer be published, dequeue them manually.

3.4 Starting the OA Adapter

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

- On UNIX, to start the OA adapter:

  1. Change to the directory containing the start script.

     cd ORACLE_HOME/oai/9.0.4/adapters/*Application*

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

- On Windows, start the OA adapter from Services.

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 is displayed.

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

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

Note: You can also start and stop the OA adapter using the IC Manager. Refer to *OracleAS InterConnect User's Guide* for more details.

3.4.1 Log File Example of Successfully Started OA Adapter

You can verify the startup status by viewing the `oailog.txt` files. These files are located in the appropriate timestamped subdirectory of the `log` directory of the OA adapter. Subdirectory names take the following form:

```
timestamp_in_milliseconds
```

The following file displays information about an OA adapter that started successfully:

```
The Adapter service is starting..
Registering your application (OAAPP)..
Initializing the Bridge oracle.oai.agent.adapter.ebusiness.EBSBridge..
Starting the Bridge oracle.oai.agent.adapter.ebusiness.EBSBridge..
oa_bridge_reader_1 has been started.
Service started successfully.
oa_bridge_writer_1 has connected to the database successfully.
oa_bridge_reader_1 has connected to the database successfully.
```

3.5 Stopping the OA Adapter

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

- On UNIX, to stop the OA adapter:

1. Change to the directory containing the stop script.

   cd ORACLE_HOME`/oai/9.0.4/adapters/`*Application*

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

- On Windows, stop the OA adapter from Services.

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 is displayed.

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

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

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

4

Sample Use Cases

This chapter describes sample use cases for the OA adapter.

4.1 Sample Use Cases for Tables/Views/APIs

4.1.1 Case One: Publish and Subscribe

This case illustrates a simple Publish-Subscribe scenario using an OA adapter at each end.

4.1.1.1 Design Time Steps

The following section describes the steps to create a Published/Subscribed Event using the Publish Wizard in iStudio:

1. Navigate to the Design tab in iStudio.
2. Invoke the Publish/Subscribe Wizard by right-clicking the appropriate node for the application.
3. In the Select an Event window choose the appropriate application and Oracle Applications as the message type. Select the event to publish/subscribe to from the list of available events. Click **Next**.
4. In the Select Interface Type window, choose the **Tables/Views/APIs** and click **Next**.
5. Model the application view in the Define the Application View window following one of the following methods:
   - Create the application view manually by clicking Add for each new attribute you would like to define.
   - Import the application view structure from one of the following sources:
     - Application Data Type

       Import an existing application data type. This option allows the re-use of data types, which have been created or imported for other Published/Subscribed Event or Invoked/Implemented Procedure before.
     - Common Data Type

       Import an existing common data type. This option may be chosen if the application view and the common view data structures are the same.
     - Tables/Views/APIs

This option allows to import the structure of tables, views, APIs and Abstract Data Types (ADTs) from an Oracle Applications database instance.

Provide the login credentials to the Oracle Applications database instance in the Database Login window. For example, Username: apps Password: apps URL: myhost.mycompany.com:1521:orcl Driver: oracle.jdbc.driver. Oracle Driver (default). Check the Save settings as Default, if you would like iStudio to remember these credentials. Click **Login**.

In the Oracle Applications Browser window, you can browse through the available Oracle Applications modules and select the structure of a table, view, API or Abstract Data Type (ADT) to define the application view. The Oracle Applications Browser also provides a search mechanism, which allows searching for table, view, API or Abstract Data Type (ADT) structures by name and/or wildcards. Enter the object name in the Object Name field. (use '%' as wildcard character, for example, HZ%V2%PUB) and check one or more of the checkboxes in order to search for table, view, API or Abstract Data Type (ADT) structures. Click **Search**. The search results will be displayed in the left window of the browser. To return to the module view, simply click **Browse**.

Select an object and click **Apply** to import the selected object as the application view.

- In the Define Mapping screen, click **New** to create transformation mappings between the Application View and Common View. Click **Next** if mappings are complete.
- If modeling a Subscribed Event, review the Generated Data Types and review/extend the generated PL/SQL package stubs in the Define Stored Procedure window.
- Click **Finish** to complete the wizard.

4.1.2 Case Two: Invoke and Implement

This case illustrates a simple Invoke-Implement scenario using an OA adapter at each end.

4.1.2.1 Design Time Steps

The following section describes the steps to create an Invoke/Implement Procedure using the Publish Wizard in iStudio:

1. Invoke the Invoke/Implement Wizard by right clicking on the appropriate node for the application.
2. In the Select a Procedure window choose the appropriate application and Oracle Applications as the message type. Select the procedure to invoke/implement from the list of available procedures. Click **Next**.
3. In the Select Interface Type window, choose **Tables/Views/APIs** and click **Next**.
4. Model the application view in the Define the Application View window following one of the following methods:
   - Create the application view manually by clicking Add for each new attribute you would like to define.
   - Import the application view structure from one of the following sources:

– Application Data Type

  Import an existing application data type. This option allows the re-use of data types, which have been created or imported for other Published/Subscribed Event or Invoked/Implemented Procedure before.

– Common Data Type

  Import an existing common data type. This option may be chosen if the application view and the common view data structures are the same.

– Tables/Views/APIs

  This option allows to import the structure of tables, views, APIs and Abstract Data Types (ADTs) from an Oracle Applications database instance.

  Provide the login credentials to the Oracle Applications database instance in the Database Login window. For example, Username: apps Password: apps URL: myhost.mycompany.com:1521:orcl Driver: oracle.jdbc.driver.OracleDriver (default). Check the Save Settings as Default if you would like iStudio to remember these credentials. Click **Login**.

  In the Oracle Applications Browser window, you can browse through the available Oracle Applications modules and select the structure of a table, view, API or Abstract Data Type (ADT) to define the application view.

  The Oracle Applications Browser also provides a search mechanism, which allows searching for table, view, API or Abstract Data Type (ADT) structures by name and/or wildcards. Enter the object name in the Object Name field. (use '%' as wildcard character, for example, HZ%V2%PUB) and check one or more of the checkboxes in order to search for table, view, API or Abstract Data Type (ADT) structures. Click **Search**. The search results will be displayed in the left window of the browser. To return to the module view, simply click **Module View**.

  Select whether you would like to import the selected object structure as IN, OUT or IN/OUT arguments.

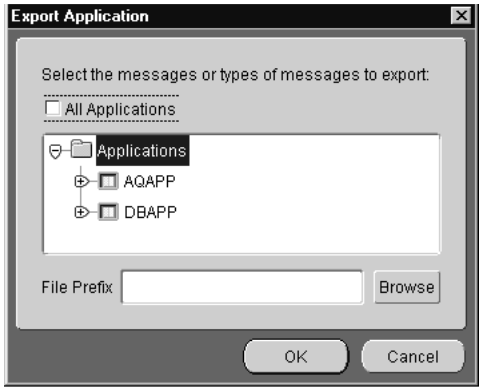
Note: If the Synchronous box is checked, then the OA adapter will function in synchronous mode.

- In the Define Mapping: IN Arguments screen, click **New** to create transformation mappings between the Application View and Common View. for the IN arguments (Request message). Click **Next** if mappings are complete.
- In the Define Mapping: OUT Arguments screen, click **New** to create transformation mappings between the Application View and Common View. for the OUT arguments (Reply message). Click **Next** if mappings are complete.
- Review the Generated Data Types and review/extend the generated PL/SQL package stubs in the Define Stored Procedure window.
- Click **Finish** to complete the wizard.

4.1.3 Runtime Steps

The runtime steps for Case One and Case Two are identical. The steps to export PL/SQL stored procedures manually:

1. Select **File** from the menu bar, then select **Export**. The Export Application dialog is displayed.



2. Select the messages to export stored procedures. Messages can be filtered as follows:
   - Export all messages—Select Applications at the top of the directory.
   - Export all messages of a certain type for all applications—Check All Applications, then select one or more types of messages to export.
   - Export all messages for a specific application—Select the application name
   - Export all messages of a certain type for a specific application—Select the type under the application name in the directory.
   - To export specific messages—Select the messages by name. To select more than one message or class of messages click the application.
3. Enter the name of the file to contain the exported stored procedures in the File Prefix field. The name generates multiple files.

   To view the directory page, click **Browse**.
4. Click **OK**. The stored procedure is now exported.
5. Install the exported stored procedures in your Oracle Applications schema. Now you're ready for runtime.

4.2 Sample Use Cases for XML Gateway

4.2.1 Case One: Publish and Subscribe

This case illustrates a simple Publish-Subscribe scenario using an OA adapter at each end.

4.2.1.1 Design Time Steps

The following section describes the steps to create a Published/Subscribed Event using the Publish Wizard in iStudio:

1. Navigate to the Design tab in iStudio.
2. Create the Application Data Type by right-clicking on the appropriate node for the application, and click **New**.

3. Enter a name for the Application Data Type. For example, adt_PROCESS_PO_007.
4. Click **Import**.
5. Select **Oracle Applications**, then click **XML Gateway**.
6. In the login dialog, give the appropriate login information to connect to the Oracle Applications instance.
   - Provide the login credentials to the Oracle Applications database instance in the Database Login window, for example
     - Username: apps
     - Password: apps
     - URL: myhost.mycompany.com: 1521: orcl
     - Driver: oracle.jdbc.driver.OracleDriver (default).
   - Check the Save settings as default, if you would like iStudio to remember these credentials.
7. After connecting successfully, the Oracle Applications Browser window will appear with all the supported DTDs for XML Gateway in the left pane.
8. Select the desired DTD. All the elements for that DTD would appear on the right pane.
9. Click **Done** to import the DTD into the Application Data Type.
10. Define the Business Object, Event and Common View, as desired.
11. Invoke the Publish/Subscribe Wizard by right-clicking the appropriate node for the application.
12. In the Select an Event window, choose the application and Oracle Applications as the message type. Select the event to. publish/subscribe to from the list of available events. Click **Next**.
13. In the Select Interface Type window, choose the **XML Gateway** radio button and click **Next**.
14. Model the application view in the Define the Application View window by importing the application view structure from one of the following sources:
    - Application Data Type

      Import an application data type as created earlier. This option allows the re-use of data types, which have been created or imported for other Published/Subscribed Event or Invoked/Implemented Procedure before.
    - Common Data Type

      Import an existing common data type. This option may be chosen if the application view and the common view data structures are the same.
    - Oracle Applications -> XML Gateway

      This option allows to import the structure of pre-existing XML Gateway transactions from an Oracle Applications database instance.

      * Login to the Oracle Applications database instance in the Database Login window.
      * In the Oracle Applications Browser window, you can browse through the available XML Gateway transactions and select the structure to define the application view.

* Select an object, and click Done to import the selected object as the application view.

15. In the Define Mapping screen, click **New** to create transformation mappings between the Application View and Common View.

> **Note:** The imported payload structure is displayed under the PAYLOAD node in the message structure.

16. Click **Next**.
17. Click **Finish** to complete the wizard.

4.2.2 Case Two: Invoke and Implement

This case illustrates a simple Invoke-Implement scenario using an OA adapter at each end.

4.2.2.1 Design Time Steps

The following section describes the steps to create a Invoke/Implement Event using the Publish Wizard in iStudio:

1. Invoke the Invoke/Implement Wizard by right-clicking on the appropriate node for the application.
2. In the Select a Procedure window choose the appropriate application and Oracle Applications as the message type. Select the procedure to invoke/implement from the list of available procedures. Click **Next**.
3. In the Select Interface Type window, choose **XML Gateway** and click **Next**.
4. Model the application view in the Define the Application View window following one of the following methods:
   - Import the application view structure from one of the following sources:
     - Application Data Type

       Import an existing application data type. This option allows the re-use of data types, which have been created or imported for other Published/Subscribed Event or Invoked/Implemented Procedure before.
     - Common Data Type

       Import an existing common data type. This option may be chosen if the application view and the common view data structures are the same.
     - XML Gateway

       This option allows to import the structure of pre-existing XML Gateway transactions from an Oracle Applications database instance

       Provide the login credentials to the Oracle Applications database instance in the Database Login window, e.g. Username: apps, Password: apps, URL: myhost.mycompany.com:1521:orcl Driver: oracle.jdbc.driver.OracleDriver (default). Check the Save Settings as Default if you would like iStudio to remember these credentials. Click **Login**.

       In the Oracle Applications Browser window, you can browse through the available XML Gateway transactions and select the structure to define the application view.

Select whether you would like to import the selected object structure as IN, OUT or IN/OUT arguments. One set of XML Gateway headers and payload should be as IN argument and another set should be OUT arguments. Depending on the scenario, the payload can also be INOUT argument type where the IN payload and the OUT payload is the same.

- In the Define Mapping: IN Arguments screen, click **New** to create transformation mappings between the Application View and Common View for the IN arguments (Request message). Click **Next** if mappings are complete.
- In the Define Mapping: OUT Arguments screen, click **New** to create transformation mappings between the Application View and Common View for the OUT arguments (Reply message). Click **Next** if mappings are complete.
- Click **Finish** to complete the wizard.

4.2.2.2 Customizing XML Gateway Payload Structures

The structure for the message payload for the XML Gateway can be customized. This is done via the Message Designer tool. This is a design time tool that is used to model the payload for the XML Gateway.

5

Frequently Asked Questions

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

5.1 Installation Questions

The following questions address installation of the OA 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 OA adapter uses this information to call a user-specified stored procedure. This user is typically the APPS user of the Oracle Applications.

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

Edit the `adapter.ini` file located in the ORACLE_HOME/oai/9.0.4/adapters/[AppType][Partition] directory.

See Also: Chapter 2, "Installation and Configuration"

How can I specify a listener port other than 1521?

Edit the oa_bridge_schema_port parameter.

See Also: Chapter 2, "Installation and Configuration"

Can I install multiple OA adapters on the same machine?

Using the Oracle Universal Installer, only one OA adapter can be installed in a single Oracle Home. However, copies of the OA adapter using the `copyAdapter` script available in the *`ORACLE_HOME`*`/oai/9.0.4/bin` directory. Usage: `copyAdapter oaapp1 oaapp2`.

The script will create a copy of the already installed OA adapter called oaapp1 with a name of oaapp2.

5.2 Design Time Questions

The following are design time questions for the OA adapter.

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

The PL/SQL code is saved 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"* in *OracleAS InterConnect Adapter for DB Installation and User's Guide*.

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

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

1. Use iStudio and select **Export** from the File menu. The Export Stored Procedures dialog is displayed.
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 5-2

Can OA messages contain arrays of arrays?

The database does not allow arrays of arrays. Therefore, the application view of OA messages should not contain arrays of arrays. For example, the application view of an OA 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.

5.3 Runtime Questions

The following questions address runtime concepts for the OA 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 OracleAS 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 is the OA adapter using old information after I changed information in iStudio?

The OA adapter caches the information from iStudio (the information which is stored in the Repository) locally for better performance in a production environment. If you change something in iStudio and want to see it in the runtime environment, stop the OA adapter, delete the cache files, and restart the adapter. Each adapter has a persistence directory located in the adapter's directory. Deleting this directory when adapter has been stopped allows the adapter to obtain the new metadata from the repository when started.

How do I find a table or a view or an API under a certain module in the Oracle Applications Browser whereas I know the object is owned by that Oracle Applications module?

Use the search facility if you are unable to find your object under the module name.

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 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 OA adapter that publishes events?

Before an OA 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 code will 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 OA 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 OA adapter that subscribes to events?

Before an OA 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 OA 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 OA adapter that implements procedures?

The steps are very similar to subscribing to events. However, the procedure that the OA 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 OA adapter, there is a way to determine whether the OA adapter was started properly. This can be viewed in the `oailog.txt` file in the logs directory of the OA adapter.

What is the consumer name?

If all the queues the OA adapter connects to on the application database side are single consumer queues, leave this blank. However, if any one of the queues is a multiconsumer queue, then specify a consumer name.

The application that writes to the OA adapter uses a consumer name to indicate to OracleAS InterConnect to pick up this message. The following two options help you to find out the consumer name to use:

- If the piece of code that writes the message to the OA adapter is already written, look at that code or the documentation that comes with it to find the consumer name.
- If the piece of code that writes the message to the OA adapter is not written, type in any string as the consumer name. When that piece of code is built, ensure that the consumer names match.

How do I handle `ANY` tags in DTDs imported into iStudio?

`ANY` tags in an XML DTD allow unstructured data in XML to be used. OracleAS InterConnect, however, must know about the structure of that data (using a DTD) if that data is to be used in mappings.

There are two methods for OracleAS InterConnect to know about the structure:

1. The simplest method is to modify the DTD being importing into iStudio and replace the `ANY` tag with structured data. When modifying the DTD, only a copy of the DTD being importing into iStudio is modified, not the published version of the DTD. For example, if the `USERAREA ANY` tag is edited before importing the DTD into iStudio, only a copy is changed and the published OAG definition which other people who download the OAG DTDs would use is not changed.

   This approach also supports using a PCDATA for an `ANY` tag.

   For example, consider the following `customer.dtd`:

   ```
   <!ELEMENT customer (name, phone, address)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT phone (#PCDATA)>
   <!ELEMENT address ANY)>
   ```

   This `customer.dtd` can be changed to the following:

   ```
   <!ELEMENT customer (name, phone, address)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT phone (#PCDATA)>
   <!ELEMENT customer (name, phone, address)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT phone (#PCDATA)>
   <!ELEMENT address (#PCDATA)>
   ```

```
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
```

This is dependent on what the XML will conform to at runtime. If the XML will use the ANY tag in different ways at runtime, a union can be used. For example, if `address` has `street`, `city`, and `state` only for some instances and for other instances only has `zip`, a standard DTD union mechanism for doing this can be used.

2. The following steps describe a second approach which involves creating a separate DTD which defines the structure used at runtime for the `ANY` tag.

   a. Import the DTD for the event, either while creating an ADT or while creating the published or subscribed event or the invoked or implemented procedure. iStudio warns about the `ANY` tag and points out the type that needs to be modified.

   b. Reload the iStudio project.

   c. Under the list of ADTs, find the type corresponding to the ANY element and right click to display the context menu. This is the ADT mentioned in step a

   d. Import a DTD which defines the structure planned to use for the `ANY` tag.

   This method does not support using a `PCDATA` tag for the `ANY` element. The ANY element must have a sub-element in this case.

   For example, consider the following `customer.dtd`:

```
<!ELEMENT customer (name, phone, address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT address ANY)>
```

   When this DTD is imported, iStudio warns that the `address` tag is an `ANY` tag and it corresponds to the `address` ADT in iStudio.

   The `address_any.dtd` could look like the following:

```
<!ELEMENT address_any (street, city, zip)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT zip ANY)>
```

   Then import the `address_any.dtd` by right-clicking on the address ADT in iStudio. This assumes the XML has an `address_any` element under the address element as follows:

```
<address>
  <address_any>
    <street>
    <city>
    <zip>
  </address_any>
</address>
```

   If the `address_any` element is not needed, then instead of editing the address ADT, edit customer ADT and change the type of address attribute from `address` to `address_any`, after importing `address_any` elsewhere. The following is now true:

```
<address>
```

```
  <street>
  <city>
  <zip>
</address>
```

Index

A

B

C

D

E

F

H

I

J

L

M

O

P

R

S

T