Oracle® Application Server Integration InterConnect

Adapter for Oracle Applications Installation and User's Guide

10*g* Release 2 (10.1.2)

B14360-02

December 2005

ORACLE®

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

B14360-02

Primary Author: Rima Dave

Contributing Author: Vimmika Dinesh

Contributor: Sivaraj Subbaiyan, Maneesh Joshi, Rahul Pathak, Harish Sriramulu

Contents

Preface v

Audience v
Documentation Accessibility v
Related Documents vi
Conventions vi

1 Introduction

1.1 Oracle Applications Overview 1-1
1.2 Oracle Applications Adapter Overview 1-1
1.2.1 Hardware Requirements 1-2
1.2.2 Software Requirements 1-2

2 Installation and Configuration

2.1 Installing the OA Adapter 2-1
2.1.1 Preinstallation Tasks 2-1
2.1.2 Installation Tasks 2-1
2.2 Installing Multiple OA Adapters in the Same Oracle Home 2-3
2.3 Configuring the OA Adapter 2-5
2.3.1 OA Adapter Ini File Settings 2-6
2.3.1.1 hub.ini Parameters 2-6
2.3.1.2 adapter.ini Parameters 2-7

3 Design-Time and Run-Time Concepts

3.1 OA Adapter Design-Time Concepts 3-1
3.2 Designing with iStudio 3-2
3.3 OA Adapter Run-Time Concepts 3-10
3.3.1 How the OA Adapter Works 3-10
3.3.1.1 OA Receiver 3-10
3.3.1.2 OA Sender 3-10
3.4 Starting the OA Adapter 3-11
3.4.1 Log File of OA Adapter 3-11
3.5 Stopping the OA Adapter 3-12

4 Sample Use Cases

4.1 Sample Use Cases for Tables/Views/APIs 4-1
4.1.1 Case One: Publish and Subscribe 4-1
4.1.1.1 Design-Time Steps 4-1
4.1.2 Case Two: Invoke and Implement 4-2
4.1.2.1 Design-Time Steps 4-2
4.1.3 Run-Time Steps 4-4
4.2 Sample Use Cases for XML Gateway 4-4
4.2.1 Case One: Publish and Subscribe 4-4
4.2.1.1 Design-Time Steps 4-4
4.2.2 Case Two: Invoke and Implement 4-6
4.2.2.1 Design-Time Steps 4-6
4.2.2.2 Customizing XML Gateway Payload Structures 4-7

A Frequently Asked Questions

A.1 General A-1
A.1.1 What should I enter on the Database User Configuration dialog box during installation? A-1
A.1.2 Is it possible to edit the database configuration settings created during installation? A-1
A.1.3 How can I specify a listener port other than 1521? A-1
A.1.4 How do I secure my passwords? A-2
A.2 Design-Time A-3
A.2.1 Where is the PL/SQL code exported through iStudio saved? A-3
A.2.2 What is the Returned IN Args feature in iStudio and how do I use it? A-3
A.2.3 How do I export stored procedures to use with the OA adapter? A-3
A.2.4 Can OA messages contain arrays of arrays? A-4
A.3 Run Time A-4
A.3.1 When I run start, I do not see anything happening: no log files are created and I do not see any messages in the console: how do I get back to the command prompt? A-4
A.3.2 Why is the OA adapter using old information after I changed information in iStudio? .. A-4
A.3.3 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? A-5
A.3.4 Why do I get errors when trying to load PL/SQL code generated through iStudio? A-5
A.3.5 What are the steps to prepare a OA adapter that publishes events? A-5
A.3.6 What are the steps to prepare a OA adapter that invokes procedures? A-6
A.3.7 What are the steps to prepare a OA adapter that subscribes to events? A-6
A.3.8 What are the steps to prepare a OA adapter that implements procedures? A-6
A.3.9 What is the consumer name? A-6
A.3.10 How do I handle ANY tags in DTDs imported into iStudio? A-7
A.3.11 My OA adapter is not starting. What could be the reason? A-8

Index

Preface

This Preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

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

- install applications
- maintain applications

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

Documentation Accessibility

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

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

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

Related Documents

For more information, refer to these Oracle resources:

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

Conventions

The following text conventions are used in this document:

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

1 Introduction

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

- Oracle Applications Overview
- Oracle Applications Adapter Overview

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 enables 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 Integration InterConnect OA adapter is required.

1.2 Oracle Applications Adapter Overview

The 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 installing the OA adapter.

Hardware	Windows	UNIX
Disk Space	400 MB	400 MB
Memory	512 MB	512 MB

1.2.2 Software Requirements

The following are software requirements for the OA adapter:

- Operating System Requirements
- JRE Requirements
- Database Requirements

Operating System Requirements

Table 1–1 lists operating system requirements for the installing the OA adapter.

Table 1–1 Operating System Requirements

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

See Also: *OracleAS Integration InterConnect Installation Guide*

JRE Requirements

OracleAS Integration InterConnect uses JRE 1.4, which is installed with its components.

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, then install Oracle8*i* or Oracle9*i* database.

Note: There is no dependency between OracleAS Integration 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 how to install and configure the OA adapter. It contains the following topics:

- Installing the OA Adapter
- Installing Multiple OA Adapters in the Same Oracle Home
- Configuring the OA Adapter

2.1 Installing the OA Adapter

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

This section contains the following topics:

- Preinstallation Tasks
- Installation Tasks

2.1.1 Preinstallation Tasks

Refer to the following guides before installing the OA adapter:

- *Oracle Application Server Installation Guide* for information about OUI startup.
- *Oracle Application Server InterConnect Installation Guide*, for information about software, hardware, and system requirements for OracleAS Integration InterConnect.

> **Note:** OracleAS Integration InterConnect hub is installable through the OracleAS Integration InterConnect hub installation type. You must install the OracleAS Integration 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 screen of the OracleAS Integration InterConnect installation Wizard, select OracleAS Integration InterConnect Adapter for Oracle Applications 10.1.2.0.2 and click **Next**.
2. The Set Oracle Wallet Password screen is displayed. Enter and confirm the password on the screen, which will be used to manage OracleAS Integration InterConnect installation. Click **Next**.

- Go to step 3 if installing the OA adapter in an OracleAS Middle Tier Oracle home that does not have an InterConnect component already installed. Ensure that the OracleAS Integration InterConnect hub has been installed.
- Go to step 4 if installing the OA adapter in an OracleAS Middle Tier Oracle home that has an existing InterConnect component. Ensure that it is a home directory to an OracleAS Integration InterConnect component.

3. The Specify Hub Database Connection screen is displayed. Enter information in the following fields:
   - Host Name: The host name of the computer where the hub database is installed.
   - Port Number: The TNS listener port for the hub database.
   - Database SID: The SID for the hub database.
   - Password: The password for the hub database user.
4. Click **Next**. The Specify Oracle Applications Adapter Name screen is displayed.
5. Enter the application name. Blank spaces are not permitted. The default value is `myOracleAppsAdapter`.

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

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

   - Section A.3.11, "My OA adapter is not starting. What could be the reason?"
   - Section A.1.4, "How do I secure my passwords?"

   ---

6. Click **Next**. The Specify Spoke Application Database Connection screen is displayed. This configures the information to the spoke application database. Enter information in the following fields:
   - Host Name: The name of the computer where the application database is installed.
   - Port Number: The TNS listener port for the application database.
   - Database SID: The SID for the application database.
   - Sys Password: The password of the sys user in the spoke database.

   The information on this screen is for Oracle Applications, from which the adapter will deliver or receive messages.
7. Click **Next**. The Specify APPS Schema Password screen is displayed. Enter the password for the schema name in Password field.
8. Click **Next**. The Set Bridge Schema Password screen is displayed.
9. Enter and confirm the password for the bridge schema.
10. Click **Next**. The Summary screen is displayed.
11. Click **Install** to install the OA adapter and other selected components. The OA adapter is installed in the following directory:

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

Application is the value specified in Step 5.

12. Click **Exit** on the Installation screen to exit the OA adapter installation.

2.2 Installing Multiple OA Adapters in the Same Oracle Home

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

Usage: copyAdapter *app1 app2*

For example, you have one instance of OA adapter with name myOAApp installed on a computer. To install another instance of the OA adapter with name myOAApp1 in the same Oracle home, use the following command:

```
copyAdapter myOAApp myOAApp1
```

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

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

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

After running the copyAdapter script, If you want to manage or monitor the newly installed adapter through Oracle Enterprise Manager 10*g* Application Server Control Console, then you need to modify the opmn.xml file by adding information about the new instance. For example, you have created a new instance of the OA adapter myOAApp1 by using the copyAdapter script. To manage the myOAApp1 adapter through Enterprise Manager, perform the following:

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

   ```
   emctl stop iasconsole
   ```

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

   ```
   <process-type id="myOAApp1" module-id="adapter" working-dir="$ORACLE_
   HOME/integration/interconnect/adapters/myOAApp1" status="enabled">
          <start timeout="600" retry="2"/>
          <stop timeout="120"/>
          <port id="icadapter_dmsport_range" range="15701-15800"/>
          <process-set id="myOAApp1" restart-on-death="true" numprocs="1">
               <module-data>
                   <category id="start-parameters">
                       <data id="java-parameters" value="-Xms8M"/>
                       <data id="class-name"
   ```

```
                    value="oracle.oai.agent.service.AgentService"/>
                </category>
                <category id="stop-parameters">
                    <data id="java-parameters" value="-mx64m"/>
                    <data id="class-name"
                     value="oracle.oai.agent.proxy.ShutdownAgent"/>
                    <data id="application-parameters"
                     value="persistence/Agent.ior"/>
                </category>
            </module-data>
       </process-set>
</process-type>
```

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

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

<process-type id="myOAApp1" module-id="adapter" working-dir="$ORACLE
_HOME/integration/interconnect/adapters/myOAApp1" status="enabled">
       <start timeout="600" retry="2"/>
       <stop timeout="120"/>
       <port id="icadapter_dmsport_range" range="15701-15800"/>
       <process-set id="myOAApp1" restart-on-death="true" numprocs="1">
        <module-data>
          <category id="start-parameters">
              <data id="java-parameters" value="-Xms8M"/>
              <data id="class-name"
               value="oracle.oai.agent.service.AgentService"/>
          </category>
          <category id="stop-parameters">
               <data id="java-parameters" value="-mx64m"/>
               <data id="class-name"
                value="oracle.oai.agent.proxy.ShutdownAgent"/>
               <data id="application-parameters"
                value="persistence/Agent.ior"/>
          </category>
        </module-data>
       </process-set>
</process-type>
```

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

   ```
   opmnctl reload
   ```

5. You can start the `myOAApp1` adapter by using the following command

   ```
   opmnctl startproc ias-component="InterConnect" process-type="myOAApp1"
   ```

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

   ```
   emctl start iasconsole
   ```

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

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

```
ApplicationName/password
```

The number of entries is dependent on the type of adapter. For example, Database Adapter needs two entries whereas AQ Adapter needs only one entry. For more information about how to manage your passwords in Oracle Wallet, refer to Section A.1.4, "How do I secure my passwords?" in Appendix A, "Frequently Asked Questions"

2.3 Configuring the OA Adapter

After an OA adapter installation, you can configure it according to your requirements. The following tables describe the location and details of the configuration files.

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

Table 2–1 OA Adapter Directory

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

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

Table 2–2 OA Executable Files

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

Table 2–3 describes the OA adapter configuration files.

Table 2–3 OA Configuration Files

File	Description
`adapter.ini` (UNIX)	Consists of all the initialization parameters that the adapter reads at startup.
`adapter.ini` (Windows)	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
`logs`	The logging of adapter activity is done in subdirectories of the logs directory. Each new run of the adapter creates a new subdirectory in which logging is done in an `log.xml` file.
`persistence`	The messages are persisted (made available) in this directory. Do not edit this directory or its files.

2.3.1 OA Adapter Ini File Settings

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

- hub.ini Parameters
- adapter.ini Parameters

2.3.1.1 hub.ini Parameters

The OA adapter connects to the hub database using the parameters in the `hub.ini` file located in the `hub` directory. Table 2–5 gives a description and an example for each parameter.

Table 2–5 hub.ini Parameters

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

Oracle Real Application Clusters hub.ini Parameters

When a hub is installed on 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 which can range from 2 to total number of nodes in cluster. For example, if the cluster contains 4 nodes, then x can be a value between 2 and 4.

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

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

2.3.1.2 adapter.ini Parameters

The agent component of the OA adapter reads the adapter.ini file at run time to access information on configuring the OA adapter parameter. Table 2–7 gives a description and an example for each parameter.

Table 2–7 adapter.ini Parameters

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

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

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

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

Parameter	Description	Example
agent_ persistence_ cleanup_interval	Specifies how often to run the persistence cleaner thread in milliseconds. Possible value: An integer greater than or equal to 30000 milliseconds Default value: 60000	agent_persistence_ cleanup_ interval=60000
agent_ persistence_ queue_size	Specifies the maximum size of internal OracleAS Integration InterConnect persistence queues. Possible value: An integer greater than or equal to 1 Default value: 1000	agent_persistence_ queue_size=1000
agent_ persistence_ retry_interval	Specifies how often the persistence thread retries when it fails to send an OracleAS Integration InterConnect message. Possible value: An integer greater than or equal to 5000 milliseconds Default value: 60000	agent_persistence_ retry_interval=60000
agent_pipeline_ from_hub	Specifies whether to activate the pipeline for messages from the hub to the bridge. If you set the pipeline to false, then the file persistence is not used in that direction. Possible value: true, false Default value: false	agent_pipeline_from_ hub=false
agent_pipeline_ to_hub	Specifies whether to activate the pipeline for messages from the bridge to the hub. If you set the pipeline to false, then the file persistence is not used in that direction. Possible value: true, false Default value: false	agent_pipeline_to_ hub=false
agent_reply_ message_selector	Specifies the application instance to which the reply must be sent. This parameter is used if multiple adapter instances exist for the given application and given partition. Possible value: A string built using the application name (parameter:application) concatenated with the instance number (parameter:instance_number). Default value: None	If application=aqapp, instance_number=2, then agent_reply_message_ selector=recipient_list like '%,aqapp2,%'
agent_reply_ subscriber_name	Specifies the subscriber name used when multiple adapter instances are used for the given application and given partition. This parameter is optional if only one instance is running. Possible value: The application name (parameter:application) concatenated with the instance number (parameter:instance_number). Default value: None	If application=oaapp and instance_ number=2, then agent_ reply_subscriber_ name=oaapp2
agent_ subscriber_name	Specifies the subscriber name used when this adapter registers its subscription. Possible value: A valid Oracle Advanced Queue subscriber name Default value: None	agent_subscriber_ name=oaapp

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

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

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

Parameter	Description	Example
nls_country	Specifies the ISO country code. The codes are defined by ISO-3166. Possible value: A valid code. A full list of the codes is available at http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html Default value: US **Note**: This parameter specifies date format and is applicable only for the date format.	nls_country=US
nls_date_format	Specifies the format for a date field expressed as a string. Possible value: A valid date format pattern as shown in Table 2–8 for the definitions of the format characters. Default value: EEE MMM dd HHmmss zzz yyyy	Date format pattern dd/MMM/yyyy can represent 01/01/2003. nls_date_format=dd-MMM-yy Multiple date formats can be specified as num_nls_formats=2 nls_date_format1=dd-MMM-yy nls_date_format2=dd/MMM/yy
nls_language	Specifies the ISO language code. The codes are defined by ISO-639. Possible value: A valid code. A full list of these codes is available at http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt Default value: en **Note**: This parameter specifies date format and is applicable only for the date format.	nls_language=en
partition	Specifies the partition this adapter handles as specified in iStudio. Possible value: An alphanumeric string Default value: None	partition=germany
service_class	Specifies the entry class for the Windows service. Possible value: oracle/oai/agent/service/AgentService. Default value: None	service_class=oracle/oai/agent/service/AgentService
service_classpath	Specifies the class path used by the adapter JVM. If a custom adapter is developed and the adapter is to pick up any additional jar files, then add the files to the existing set of jar files. Possible value: A valid PATH setting Default value: None This parameter is only for Microsoft Windows.	service_classpath=D:\oracle\oraic\integration\interconnect\lib\oai.jar;D:\oracle\oraic\jdbc\classes12.zip

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

Parameter	Description	Example
service_jdk_dll	Specifies the Dynamic Link Library(DLL) that the adapter JVM should use. Possible value: A valid jvm.dll Default value: jvm.dll This parameter is only for Microsoft Windows.	service_jdk_dll=jvm.dll
service_jdk_version	Specifies the JDK version that the adapter JVM should use. Possible value: A valid JDK version number Default value: 1.4 This parameter is only for Microsoft Windows.	service_jdk_version=1.4
service_max_heap_size	Specifies the maximum heap size for the adapter JVM. Possible value: A valid JVM heap size Default value: 536870912 This parameter is only for Microsoft Windows.	service_max_heap_size=536870912
service_max_java_stack_size	Specifies the maximum size to which the JVM stack can grow. Possible value: A valid JVM maximum stack size Default value: Default value for the JVM This parameter is only for Microsoft Windows.	service_max_java_stack_size=409600
service_max_native_stack_size	Specifies the maximum size to which the JVM native stack can grow. Possible value: The valid JVM maximum native stack size Default value: Default value for the JVM This parameter is only for Microsoft Windows.	service_max_native_size=131072
service_min_heap_size	Specifies the minimum heap size for the adapter JVM. Possible value: The valid JVM heap size Default value: 536870912 This parameter is only for Microsoft Windows.	service_min_heap_size=536870912

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

Parameter	Description	Example
service_num_vm_args	Specifies the number of service_vm_arg*number* parameters specified in JVM. Possible value: The number of service_vm_arg*number* parameters Default value: None This parameter is only for Microsoft Windows.	service_num_vm_args=1
service_path	Specifies the environment variable PATH. The PATH variable is set before starting the Java Virtual Machine (JVM). Typically, list all directories that contain necessary DLLs. Possible value: The valid PATH environment variable setting Default value: None This parameter is only for Microsoft Windows.	service_path=%JREHOME%\bin;D:\oracle\oraic\bin
service_vm_arg*number*	Specifies any additional arguments to the JVM. For example, to retrieve line numbers in any stack traces, set service_vm_arg1=java.compiler=NONE. If a list of arguments exists, then use multiple parameters as shown in the example, by incrementing the last digit by 1. Possible value: A valid JVM arguments Default value: None This parameter is only for Microsoft Windows.	service_vm_arg1=java.compiler=NONE service_vm_arg2=oai.adapter=.aq

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

Table 2–8 Reserved Characters for the nls_date_format Parameter

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

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

Letter	Description	Example
s	Second in minute	55
S	Millisecond	978

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. Default value: oracle.oai.agent.adapter.ebs.EBSBridge	bridge_class=oracle.oai.agent.adapter.ebs.EBSBridge
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. Possible value: ebs_bridge_username Default value: None	ebs_aq_bridge_consumer_name=ebsuser
ebs_aq_bridge_owner	Specifies application (spoke) database queue (applicable for XML Gateway, WF Queue, and Custom Queue only). Default value: None	ebs_aq_bridge_owner=apps
ebs_bridge_schema_host	Specifies the name of the computer hosting the database instance. Possible value: The name of the computer hosting the database Default value: None	ebs_bridge_schema_host=dlsun4255
ebs_bridge_schema_instance	Specifies the SID of the database instance. Default value: Non.	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 and each thread has its own database session. Possible values: An integer greater than 0 Default value: None	ebs_bridge_schema_num_readers=1
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 and each thread has its own database session. Possible values: An integer greater than 0 Default value: None	ebs_bridge_schema_num_writers=1

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

Parameter	Description	Example
ebs_bridge_ schema_password	Specifies the password for the user in the ebs_ bridge_schemas*schema*#_username. Possible value: The password for the corresponding database user Default value: None **Note**: All passwords are stored in Oracle Wallet. Refer to Section A.1.4, "How do I secure my passwords?" in Appendix A, "Frequently Asked Questions" for more details on how to modify and retrieve the password using Oracle Wallet.	ebs_bridge_schema1_ password=oai encrypted_ebs_bridge_ schema1_ password=112511011064 109110871093
ebs_bridge_ schema_port	Specifies the port where the TNS listener is running for the database instance. Possible value: A valid TNS listener port number Default value: None	ebs_bridge_schema_ port=1521
ebs_bridge_ schema_username	Specifies the user name 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: A valid database user name Default value: None	ebs_bridge_schema1_ username=oai
ebs_bridge_ schema_writer_ password	Specifies the password corresponding to the database. Default value: None	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. Possible values: true or false The default value is false.	ebs_bridge_schema1_ writer_use_oracle_ objects=true
ebs_bridge_ schema_writer_ username	Specifies the user name to be used by this writer to log on to the database. Default value: None	ebs_bridge_schema_ writer_username=apps
ebs_bridge_sql_ trace	Specifies whether to turn sql tracing on for the APPS database. Possible values: true or false Default value: false	ebs_bridge_sql_trace =true
ebs_bridge_use_ thin_jdbc	Specifies whether to use a thin JDBC driver when talking to the APPS database. Possible values: true or false Default value: true	EBS_bridge_thin_ jdbc=true

3

Design-Time and Run-Time Concepts

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

- OA Adapter Design-Time Concepts
- Designing with iStudio
- OA Adapter Run-Time Concepts
- Starting the OA Adapter
- Stopping the OA Adapter

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 is used to integrate with Oracle Applications through direct database access using tables/views/APIs/ADTs. This interface type enables 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 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 enables 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 OracleAS Integration 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 enables 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 enables 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.

  While using the WF BES Interface, set the recipient list on the message that you are enqueuing on to the BES queue, by using the AddHeader transformation to create an `aq_recipients` field in the message header. Then, add the recipients to the aq_recipients field. OracleAS Integration InterConnect AQ adapter will automatically use the contents of aq_recipients as the recipient list for the message.

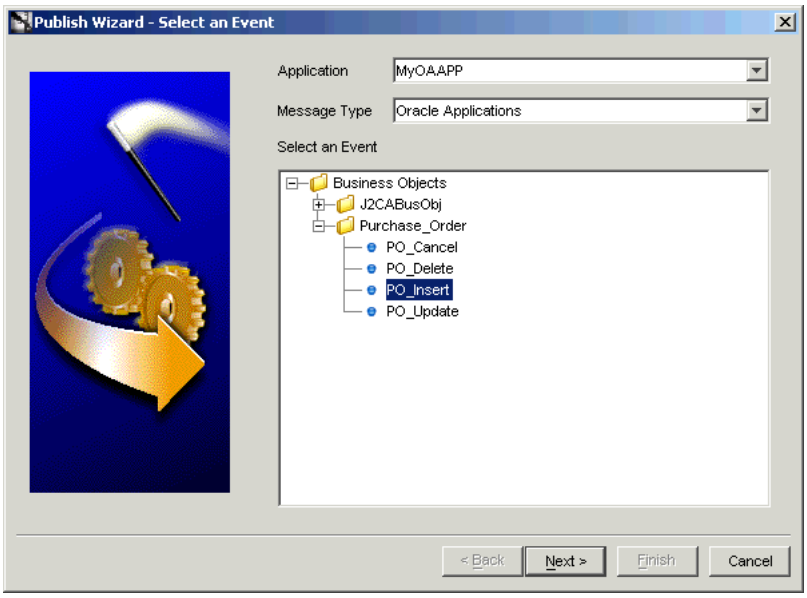
- Custom Queues

  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 enables 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.

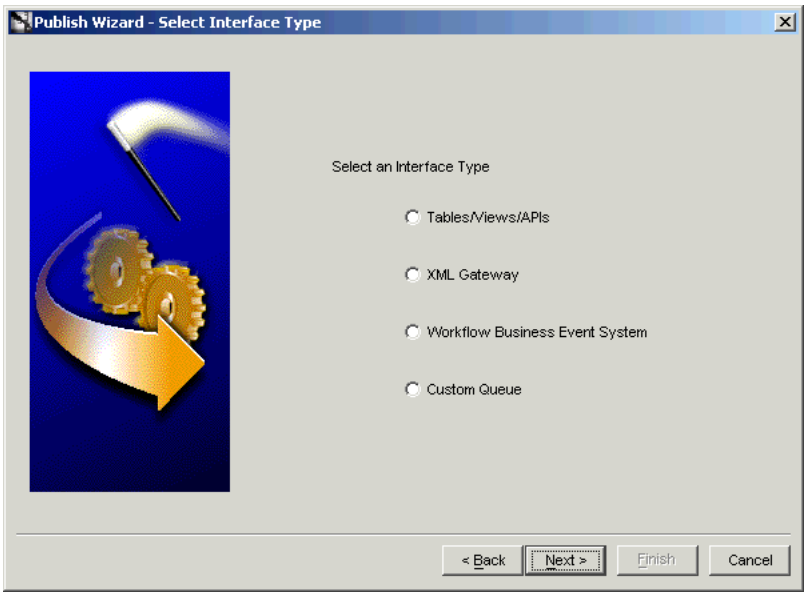
3.2 Designing with iStudio

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

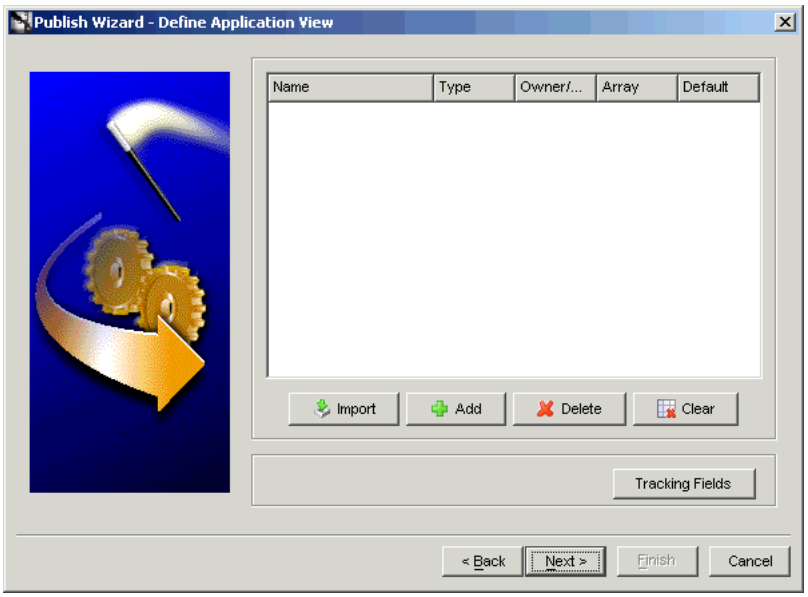
1. In the Design Navigation list, expand the **Application** node.
2. Right-click **Published Events** and select **New**. The Publish Wizard is displayed.



3. 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 Integration InterConnect and the application. Select the Oracle Applications message type.
4. Select the event name.
5. Click **Next**. The Select Interface Type dialog box is displayed.
6. 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

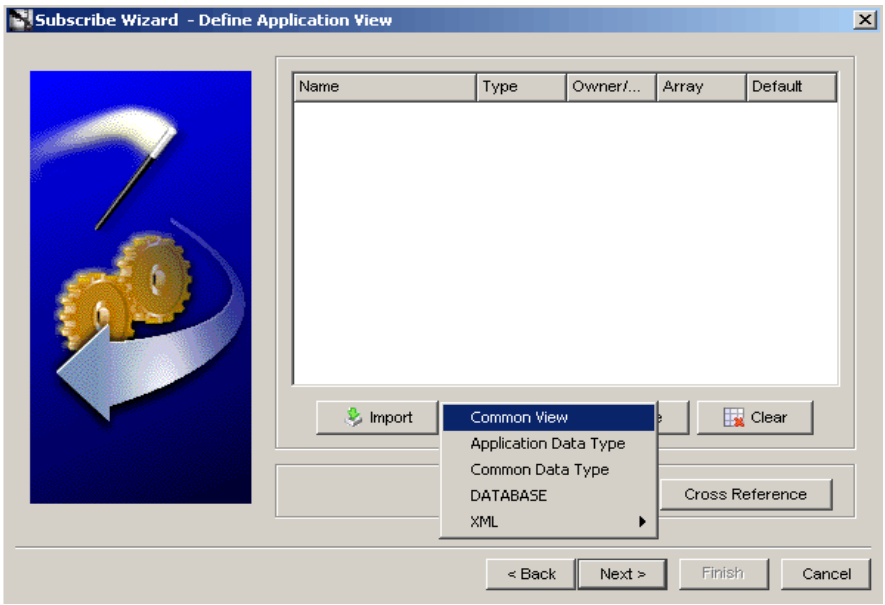


7. Click **Next**. The Define Application View dialog box is displayed.



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

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



Depending on the option selected in Step 6, the import menu options vary, as displayed in Table 3–1.

Table 3–1 Import Menu Options

If you had selected...	Import Menu Options are...
Tables/Views/APIs	Common View, Application Data Type, Common Data Type, Tables/Views/APIs

Table 3–1 (Cont.) Import Menu Options

If you had selected...	Import Menu Options are...
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

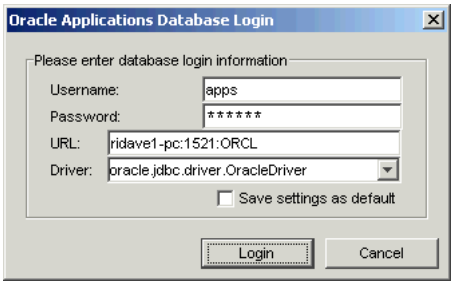
If you select Common View, Application Data Type, or Common Data Type, then the steps to be followed are identical to those explained in the *Oracle Application Server Integration InterConnect User's Guide*.

If you select the XML option from the Import Menu, then go to Step 11.

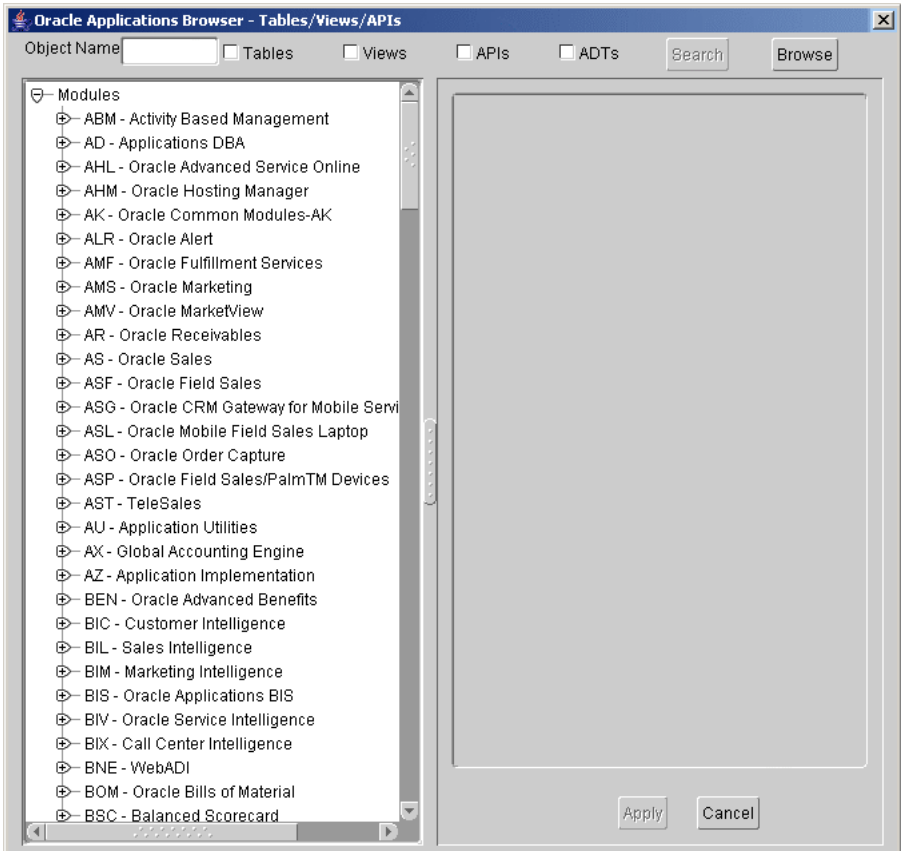
If you select Tables/Views/APIs, then the Oracle Applications Login dialog box is displayed as shown in Step 9.

9. Enter information in the following fields:

- User Name: The database log in name. By default, the user name for the application schema of the Oracle Applications is apps.
- Password: The database log in password.
- URL: The computer 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.



10. 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:

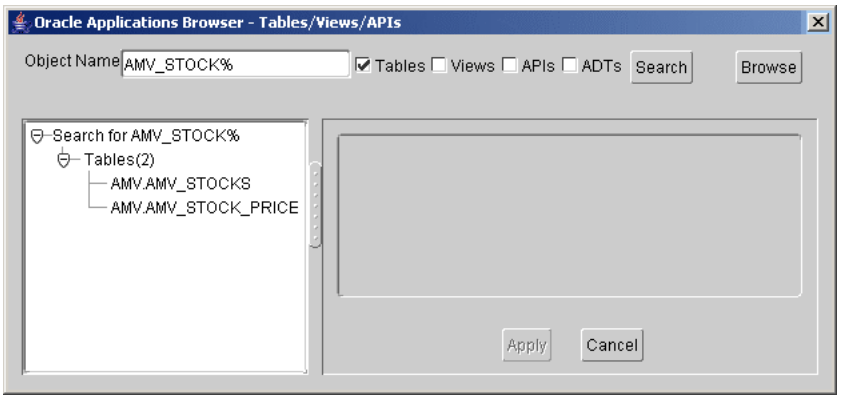
- Browse Modules: Browse through the list of modules displayed, and select the desired object. The attributes of the selected object are displayed on the right pane of the Browser. Select the desired attributes, and click Apply. The Define Application View dialog box is displayed with the selected attributes. Go to Step 14 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", then AR_*package name* packages only will be displayed under this module. If the Oracle Receivables module has RA_*package name* and HZ_*package name* packages also, then 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 required object types (Tables, Views, APIs, ADTs).

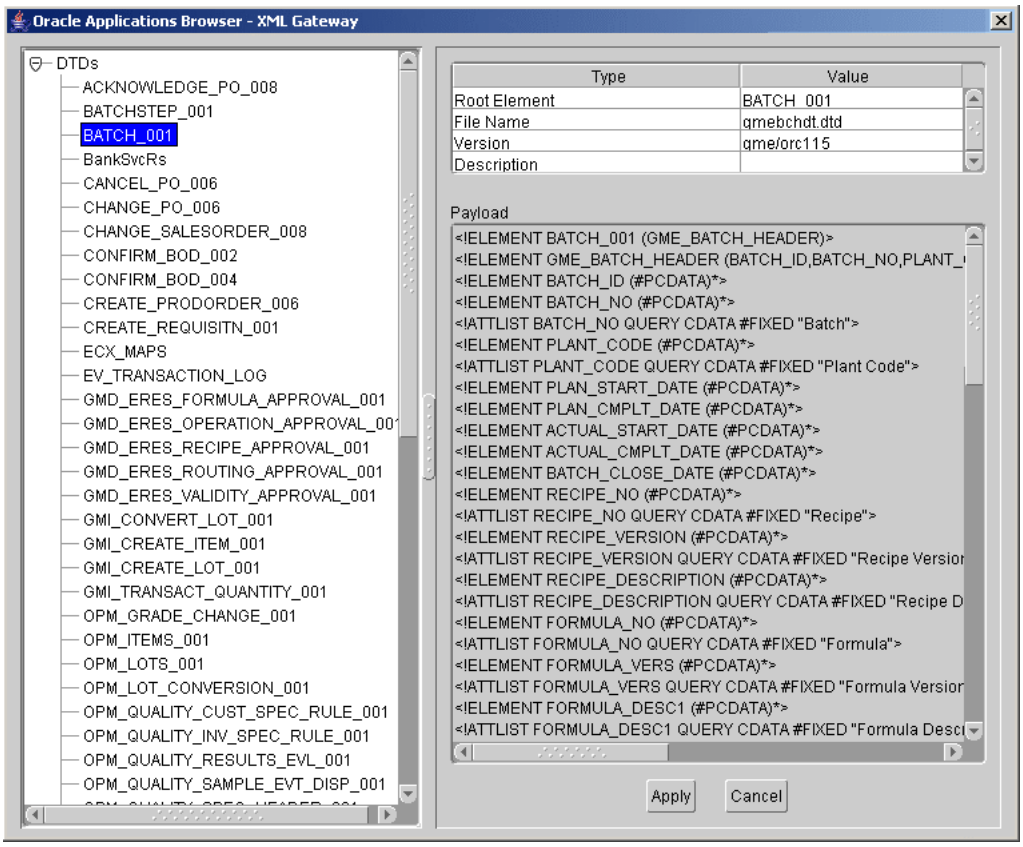


If you are not sure of the complete name of the desired object, then you can enter the entire object name that you are looking for or use the wild card '%', for example, `AMV_STOCK%`. You must select at least one of the database object types listed at the top of the dialog box to enable the Search button.

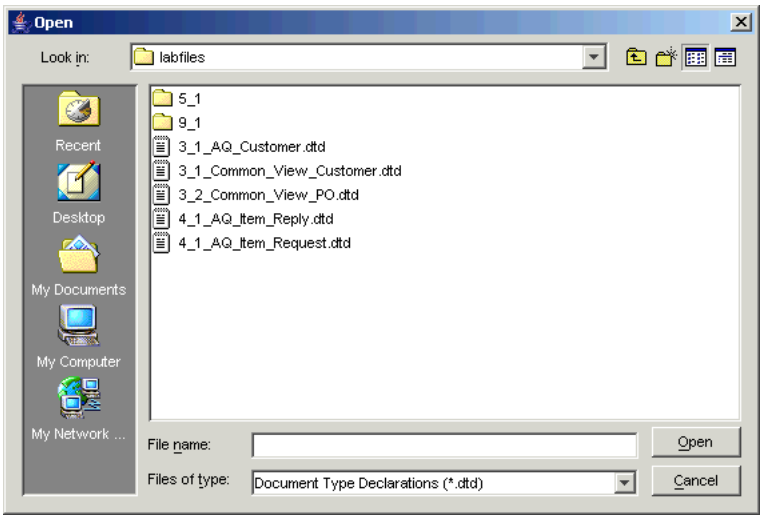
Click Search to display the list of matching object names. Select the desired object, the attributes of the selected object are displayed on the right pane of the Browser. Select the desired attributes, and click Apply. The Define Application View dialog box is displayed with the selected attributes. Go to Step 14 to continue.

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

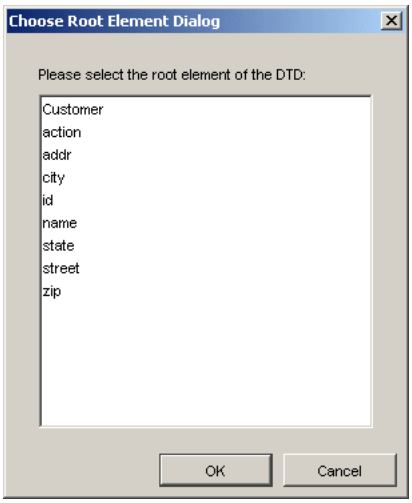
If you have selected XML Gateway in Step 6, then the XML Gateway Browser is displayed. In the Browser, the left pane displays a list of DTDs and the right pane displays the details of the DTD such as its root element, file name, version, description, and payload. Also, the payload details of the selected DTD are displayed, as shown in the following figure. Select a DTD, and click Apply. The Define Application View dialog box is displayed with the selected attributes. Go to Step 14 to continue.



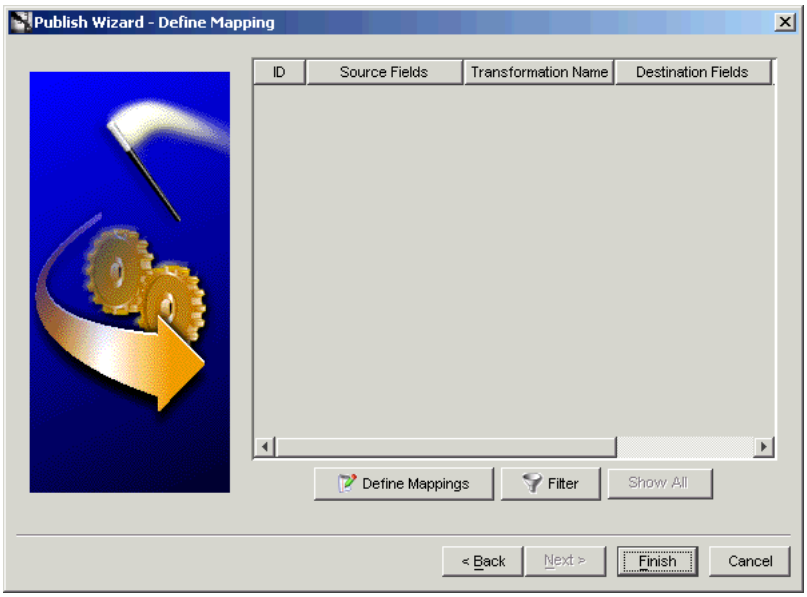
11. The Open dialog box is displayed. Select a DTD file, and click **Open**.



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



13. Click **OK**. The Define Application View dialog box is displayed with the imported DTD.
14. 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.



15. Click **Define Mappings** to define new mappings. The Mapping Parameters dialog box 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.
16. Click **OK** to return to the Publish Event Wizard.
17. Click **Finish**.

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 Run-Time Concepts

This section describes the key run-time components of the OA 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 for each interface type.

3.3.1 How the OA Adapter Works

This section contains the following 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 required 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.

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

3.3.1.2 OA Sender

The OA adapter is comprised of the bridge and the run-time 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 Integration InterConnect message waiting to be sent by this adapter. The adapter then picks up the message from the interface tables residing in the oai schema, builds the corresponding OracleAS Integration InterConnect message, persists it, transforms it to the common view, and routes it to the hub. From the hub, the message gets routed to the required 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 Integration 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, then 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 queues (XML Gateway, Workflow BES, Custom Queue) selected for publishing messages in the `apps` schema. A new message in this queue indicates a new outbound OracleAS Integration InterConnect message waiting to be sent by the adapter. The adapter then picks up the message, builds the corresponding OracleAS Integration InterConnect message, persists it, transforms it to the common view, and routes it to the hub. From the hub, the message is routed to the required subscriber based on configuration done using iStudio, which could be content-based, or subscription-based.

  The application and the OA adapter communicate through the publishing and invoking queues, residing in the `apps` schema for outbound messages and through subscribing and implementing queues for inbound messages. Therefore, the OA adapter is down while the application is publishing OracleAS Integration 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 starting the adapter varies.

- To start the OA adapter on UNIX:
  1. Change to the directory containing the start script.

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

  2. Type **`start`** and press **Enter**.
- To start the OA adapter from Services on Windows:
  1. Access the Services window from the Start menu.
  2. Select the ***OracleHomeOracleASInterConnectAdapter-Application*** service.
  3. Start the service.

> **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 of OA Adapter

You can verify the start up status by viewing the `log.xml` files. The files are located in the time-stamped subdirectory of the `log` directory of the OA adapter. Subdirectory names take the following form:

```
timestamp_in_milliseconds
```

The following is an example of the 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.

- To stop the OA adapter on UNIX:
  1. Change to the directory containing the stop script.

     cd ORACLE_HOME/integration/interconnect/adapters/*Application*

  2. Type **`stop`** and press **Enter**.
- To stop the OA adapter from Services on Windows:
  1. Access the Services window from the Start menu.
  2. Select the ***OracleHomeOracleASInterConnectAdapter-Application*** service.
  3. Stop the service. Based on the operating system, the method for stopping it varies.

  You can verify the stop status of the OA adapter by viewing the `log.xml` files. These files are located in the time-stamped subdirectory of the `log` directory of the OA adapter.

4

Sample Use Cases

This chapter describes sample use cases for the OA adapter.

- Sample Use Cases for Tables/Views/APIs
- Sample Use Cases for XML Gateway

4.1 Sample Use Cases for Tables/Views/APIs

This section contains the following use cases:

- Case One: Publish and Subscribe
- Case Two: Invoke and Implement

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 node for the application.
3. In the Select an Event dialog box select the required 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 dialog box, select the **Tables/Views/APIs** and click **Next**.
5. Model the application view in the Define the Application View dialog box 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 enables the reuse 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 selected if the application view and the common view data structures are the same.

– Tables/Views/APIs

  This option enables you 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 dialog box. For example, `Username: apps Password: apps URL: myhost.mycompany.com:1521:orcl Driver: oracle.jdbc.driver. Oracle Driver (default).` Select the **Save settings as Default**, if you would like iStudio to remember these credentials. Click **Login**.

  In the Oracle Applications Browser dialog box, 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 enables searching for table, view, API or Abstract Data Type (ADT) structures by name and 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, click **Browse**.

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

– In the Define Mapping dialog box, click **Define Mappings** 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 dialog box.

– 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. Right-click the required node for the application to invoke the Invoke/Implement Wizard.
2. In the Select a Procedure dialog box, select the required application and **Oracle Applications** as the message type.

3. Select the procedure to invoke/implement from the list of available procedures. Click **Next**.
4. In the Select Interface Type dialog box, select **Tables/Views/APIs** and click **Next**.
5. Model the application view in the Define the Application View dialog box by using 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 enables the reuse 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 can be selected if the application view and the common view data structures are the same.

     – Tables/Views/APIs

       This option enables importing 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 dialog box. For example, `Username: apps Password: apps URL: myhost.mycompany.com:1521:orcl Driver: oracle.jdbc.driver.OracleDriver (default)`. Select the **Save Settings as Default** if you would like iStudio to remember these credentials. Click **Login**.

       In the Oracle Applications Browser dialog box, 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 enables searching for table, view, API or Abstract Data Type (ADT) structures by name and 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 dialog box 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 dialog box, click **Define Mappings** 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 dialog box, 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 dialog box.
- Click **Finish** to complete the wizard.

4.1.3 Run-Time Steps

The run-time 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 box 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 are ready for run time.

4.2 Sample Use Cases for XML Gateway

This section contains the following use cases:

- Case One: Publish and Subscribe
- Case Two: Invoke and Implement

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. Right-click the node for the application to create the Application Data Type, 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 box, give the required login information to connect to the Oracle Applications instance.
   - Provide the login credentials to the Oracle Applications database instance in the Database Login dialog box, for example
     - Username: `apps`
     - Password: `apps`
     - URL: `myhost.mycompany.com: 1521: orcl`
     - Driver: `oracle.jdbc.driver.OracleDriver (default)`
   - Check the **Save settings as default** option, if you would like iStudio to remember these credentials.
7. After connecting successfully, the Oracle Applications Browser dialog box is displayed with all the supported DTDs for XML Gateway in the left pane.
8. Select the desired DTD. All the elements for that DTD are displayed in the right pane.
9. Click **Done** to import the DTD into the Application Data Type.
10. Define the Business Object, Event and Common View.
11. Right-click the node for the application to invoke the Publish/Subscribe Wizard.
12. In the Select an Event dialog box, select 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 dialog box, select the **XML Gateway** and click **Next**.
14. Model the application view in the Define the Application View dialog box 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 enables the reuse 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 selected if the application view and the common view data structures are the same.
    - Oracle Applications: XML Gateway

      This option enables 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 dialog box.

In the Oracle Applications Browser dialog box, 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 dialog box, click **Define Mappings** 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 the node for the application.
2. In the Select a Procedure dialog box, select the required 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 dialog box, select **XML Gateway** and click **Next**.
4. Model the application view in the Define the Application View dialog box 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 enables the reuse 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 selected if the application view and the common view data structures are the same.
     - XML Gateway

       This option enables 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 dialog box, 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 dialog box, 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 dialog box, click **Define Mappings** 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 dialog box, click **Define Mapping** 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 using the Message Designer tool. This is a design-time tool that is used to model the payload for the XML Gateway.

A

Frequently Asked Questions

This chapter provides answers to frequently asked questions about the OA adapter. It contains frequently asked questions for following categories:

- General
- Design-Time
- Run Time

A.1 General

This sections explains following frequently asked questions about the OA adapter:

- What should I enter on the Database User Configuration dialog box during installation?
- Is it possible to edit the database configuration settings created during installation?
- How can I specify a listener port other than 1521?
- How do I secure my passwords?

A.1.1 What should I enter on the Database User Configuration dialog box during installation?

This information is used to find where the stored procedures generated through iStudio will be installed for application inbound messages. At run time, the OA adapter uses this information to call a user-specified stored procedure. This user is typically the APPS user of the Oracle Applications.

A.1.2 Is it possible to edit the database configuration settings created during installation?

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

See Also: Chapter 2, "Installation and Configuration"

A.1.3 How can I specify a listener port other than 1521?

Edit the oa_bridge_schema_port parameter.

See Also: Chapter 2, "Installation and Configuration"

A.1.4 How do I secure my passwords?

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

```
ApplicationName/password
```

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

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

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

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

You can use the following commands to manage your passwords:

- List all passwords in the store

  ```
  oraclewallet -listsecrets
  ```

- Create a password

  ```
  oraclewallet -createsecret passwordname
  ```

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

  ```
  oraclewallet -createsecret hub_password
  ```

- View a password

  ```
  oraclewallet -viewsecret passwordname
  ```

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

  ```
  oraclewallet -viewsecret hub_password
  ```

- Update a password

  ```
  oraclewallet -updatesecret passwordname
  ```

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

  ```
  oraclewallet -updatesecret hub_password
  ```

- Delete a password

  ```
  oraclewallet -deletesecret passwordname
  ```

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

  ```
  oraclewallet -deletesecret hub_password
  ```

A.2 Design-Time

Following design time related frequently asked questions are explained:

- Where is the PL/SQL code exported through iStudio saved?
- What is the Returned IN Args feature in iStudio and how do I use it?
- How do I export stored procedures to use with the OA adapter?
- Can OA messages contain arrays of arrays?

A.2.1 Where is the PL/SQL code exported through iStudio saved?

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

PrefixSpecifiedInIStudio_ApplicationName_BusinessObject`TYPES.sql`
PrefixSpecifiedInIStudio_ApplicationName_BusinessObject`.sql`

A.2.2 What is the Returned IN Args feature in iStudio and how do I use it?

Please refer to *"Returned In Arguments"* in *Oracle Application Server Integration InterConnect User's Guide*.

A.2.3 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 list 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, then select one or more applications. Only the stored procedures for those applications will be generated. You can also select messages based on the role. For

example, if you select publish, then only publish messages will be generated. Or, you can select to export the stored procedures for specific messages by selecting those messages in the list.

A.2.4 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.

A.3 Run Time

Following run time related frequently asked questions are explained:

- When I run start, I do not see anything happening: no log files are created and I do not see any messages in the console: how do I get back to the command prompt?
- Why is the OA adapter using old information after I changed information in iStudio?
- 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?
- Why do I get errors when trying to load PL/SQL code generated through iStudio?
- What are the steps to prepare a OA adapter that publishes events?
- What are the steps to prepare a OA adapter that invokes procedures?
- What are the steps to prepare a OA adapter that subscribes to events?
- What are the steps to prepare a OA adapter that implements procedures?
- What is the consumer name?
- How do I handle ANY tags in DTDs imported into iStudio?
- My OA adapter is not starting. What could be the reason?

A.3.1 When I run start, I do not see anything happening: no log files are created and I do not see any messages in the console: how do I get back to the command prompt?

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

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

A.3.2 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 view it in the run-time environment, then

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.

A.3.3 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.

A.3.4 Why do I get errors when trying to load PL/SQL code generated through iStudio?

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

A.3.5 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.

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

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

Next, populate the message with the correct data. For each nonprimitive 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
```

A.3.6 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.

A.3.7 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.

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.

A.3.8 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, then there is a way to determine whether the OA adapter was started properly. This can be viewed in the `log.xml` file in the logs directory of the OA adapter.

A.3.9 What is the consumer name?

If all the queues the OA adapter connects to on the application database side are single consumer queues, then 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 Integration 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, then 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, then type in any string as the consumer name. When that piece of code is built, ensure that the consumer names match.

A.3.10 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 Integration 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 Integration 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 run time. If the XML will use the ANY tag in different ways at run time, 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 run time 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 subelement 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>
```

A.3.11 My OA adapter is not starting. What could be the reason?

One reason can be that Oracle Wallet does not contain the password information corresponding to your application name. For example, during installation you defined the application name as myMQApp. Later, you changed the application name in iStudio to MQApp. In such case, you need to specify the password corresponding to the new application name MQApp in the Oracle Wallet. You can create password by using the oraclewallet command.

See Also: Section A.1.4, "How do I secure my passwords?"

Index

A

adapters
  multiple adapters in same Oracle Home, 2-3
agent
  configuration parameters, 2-7

C

Configuring the OA adapter, 2-5
copyAdapter script, 2-3
Custom Queues, 3-2

D

Database Access
  Interface Type - Tables/Views/APIs, 3-10
Database Requirements, 1-2

F

FTP adapter
  installing multiple adapters, 2-3

H

Hardware Requirements, 1-2

I

Ini File Settings, 2-6
Installation Tasks, 2-1

J

JRE Requirements, 1-2

L

Log File of OA Adapter, 3-11

M

Message Queues, 3-11

O

OA Adapter Design Time Concepts, 3-1
OA Adapter-specific Parameters, 2-14
OA Receiver, 3-10
OA Sender, 3-10
Operating System Requirements, 1-2
Oracle Applications, 1-1
Oracle Applications Adapter Overview, 1-1
Oracle Applications Overview, 1-1
Oracle Real Application Clusters hub.ini
    Parameters, 2-6

P

password security, A-2
PL/SQL code, A-3

S

securing passwords, A-2
  with Oracle wallet, A-2
Software Requirements, 1-2
specifying listener port, A-1
start (UNIX), 2-5
stop (UNIX), 2-5

T

Tables/Views/APIs, 3-1
The Select Interface dialog box is displayed., 3-3

W

Workflow Business Event System (BES), 3-2

X

XML Gateway, 3-1