

**Oracle® Application Server Integration
InterConnect**

User's Guide

10g Release 2 (10.1.2)

Part No. B14069-01

November 2004

Oracle Application Server Integration InterConnect User's Guide 10g Release 2 (10.1.2)

Part No. B14069-01

Copyright © 2003, 2004, Oracle. All rights reserved.

Primary Author: Pradeep Vasudev, Vimmy K Raj

Contributor: Rahul Pathak, Maneesh Joshi, Harish Sriramulu, Aska Onoda, Aruna Kasturi

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

Send Us Your Comments	xi
Preface	xiii
Audience	xiii
Documentation Accessibility	xiii
Structure	xiv
Related Documents	xv
Conventions	xv
1 Getting Started with OracleAS Integration InterConnect	
1.1 What is OracleAS Integration InterConnect?	1-1
1.1.1 OracleAS Integration InterConnect Components	1-2
1.1.1.1 OracleAS Integration InterConnect Hub	1-2
1.1.1.2 OracleAS Integration InterConnect Adapters	1-2
1.1.1.3 OracleAS Integration InterConnect Development Kit	1-3
1.1.1.3.1 OracleAS Integration InterConnect SDKs	1-3
1.2 Standard Messaging	1-4
1.2.1 Supported Messaging Paradigms	1-4
1.3 OracleAS Integration InterConnect Integration Process	1-5
1.3.1 Design Time	1-5
1.3.2 Runtime	1-6
1.3.3 Separation of Integration Logic and Platform Functionality	1-6
1.3.4 Unique Integration Methodology	1-6
1.3.4.1 How the Hub-and-Spoke Methodology Works	1-6
1.3.5 Integration Lifecycle Management	1-8
1.3.6 Using Adapters for Integration	1-8
1.4 What's New in This Release?	1-10
2 Using iStudio	
2.1 Overview of iStudio	2-1
2.1.1 iStudio Concepts	2-1
2.1.1.1 Applications	2-2
2.1.1.2 Common Views and Business Objects	2-2
2.1.1.2.1 Events	2-2
2.1.1.2.2 Procedures	2-3

2.1.1.3	Transformations or Mappings	2-3
2.1.1.4	Metadata Versioning	2-4
2.1.1.5	Tracking Fields	2-4
2.1.1.6	Content-Based Routing	2-4
2.1.1.7	Cross Reference Tables	2-5
2.1.1.8	Domain Value Mapping	2-5
2.1.1.9	Routing and the Message Capability Matrix	2-5
2.2	Starting iStudio	2-5
2.3	Parts of the iStudio Window	2-5
2.3.1	Menu Bar	2-6
2.3.1.1	File Menu	2-6
2.3.1.2	Edit Menu	2-7
2.3.1.3	Procedure Menu	2-7
2.3.1.4	Event Menu	2-7
2.3.1.5	Help Menu	2-8
2.3.2	Toolbar	2-8
2.3.3	Design Navigation List	2-9
2.3.4	Deploy Navigation List	2-9
2.3.5	Context Menus	2-9
2.3.6	Detail View	2-10
2.4	Using Workspaces in iStudio	2-10
2.4.1	Creating a New Workspace	2-10
2.4.2	Opening an Existing Workspace	2-10
2.5	Using Projects in iStudio	2-11
2.5.1	Creating a New Project	2-11
2.5.2	Opening an Existing Project	2-12

3 Creating Applications, Common Views, and Business Objects

3.1	Overview of Applications	3-1
3.1.1	Application View	3-1
3.1.2	Application Data Types	3-1
3.1.3	Creating an Application	3-1
3.2	Overview of Common Views and Business Objects	3-2
3.2.1	Defining Common Views	3-2
3.2.1.1	Creating Business Objects	3-2
3.2.1.2	Creating Common Data Types	3-2
3.2.1.3	Adding Attributes	3-3
3.2.1.4	Importing Attributes	3-4
3.2.1.5	Deleting and Clearing Attributes	3-5

4 Using Events in iStudio

4.1	Overview of Events	4-1
4.1.1	Event Maps	4-2
4.2	Creating Events	4-2
4.3	Publishing and Subscribing to an Event	4-3
4.3.1	Publishing an Event	4-3
4.3.2	Subscribing to an Event	4-8

5 Using Procedures in iStudio

5.1	Using Procedures	5-1
5.1.1	Creating a Procedure.....	5-2
5.2	Invoking and Implementing a Procedure	5-3
5.2.1	Invoking a Procedure.....	5-3
5.2.2	Implementing a Procedure.....	5-6

6 Enabling Infrastructure

6.1	Enabling Infrastructure	6-1
6.2	Working with Content-based Routing.....	6-2
6.3	Working with Domain Value Mappings.....	6-5
6.3.1	Adding Applications to Domain Value Mappings.....	6-5
6.3.2	Removing Applications From Domain Value Mappings	6-5
6.3.3	Modifying Domain Value Mappings.....	6-6
6.3.4	Deleting Domain Value Mappings	6-6
6.3.5	Deleting Domain Value Mapping Tables.....	6-6
6.3.6	Modifying Attribute Mappings	6-7
6.3.7	Removing Attribute Mappings.....	6-7
6.4	Working with Cross-referencing	6-7
6.4.1	Adding Applications to Cross-reference Tables	6-7
6.4.2	Removing Applications From Cross-reference Tables.....	6-7
6.4.3	Populating Cross-reference Tables.....	6-7

7 Using Oracle Workflow

7.1	Oracle Workflow Overview	7-1
7.1.1	Oracle Workflow Solves Common Business Problems.....	7-1
7.1.1.1	Error Management and Compensating Transactions	7-1
7.1.1.2	Human Interaction	7-2
7.1.1.3	Message Junctions	7-2
7.1.1.4	Stateful Routing	7-2
7.1.1.5	Composite Services	7-2
7.2	OracleAS Integration InterConnect Integration with Oracle Workflow	7-3
7.2.1	Design Time Tools	7-3
7.2.2	Runtime.....	7-5
7.3	Using Oracle Workflow with OracleAS Integration InterConnect.....	7-6
7.3.1	Model Business Process	7-6
7.3.2	Deploy Business Processes for Runtime.....	7-6
7.4	Model Business Process	7-6
7.4.1	Process Bundle	7-6
7.4.2	Business Process.....	7-7
7.4.3	Activity	7-7
7.4.4	Creating a Process Bundle.....	7-8
7.4.5	Creating a Business Process	7-8
7.4.6	Populating a Business Process with Activities.....	7-8
7.4.7	Deploying to Oracle Workflow	7-9
7.4.8	Launching Oracle Workflow Tools.....	7-11

7.4.8.1	Launching the Oracle Workflow Home Page	7-12
7.4.8.2	Launching Oracle Workflow Builder	7-12
7.4.9	Modifying Existing Oracle Workflow Processes.....	7-13

8 Deployment

8.1	Deploying PL/SQL Stored Procedures	8-1
8.1.1	Manual Deployment.....	8-1
8.1.2	Auto Deployment	8-2
8.2	Specifying Application Queue Names for AQ Adapter	8-6
8.3	Deploying Workflow Events and Process Definitions	8-7
8.4	Sync Adapters from iStudio	8-7

9 Runtime System Concepts and Components

9.1	Integration Architecture.....	9-1
9.2	Components.....	9-2
9.2.1	Adapters.....	9-2
9.2.1.1	Agent and Bridge Combination	9-2
9.2.2	Repository	9-3
9.2.3	Advanced Queues	9-4
9.2.4	Oracle Workflow.....	9-4
9.3	Runtime System Features	9-4
9.3.1	Messaging Paradigms	9-4
9.3.2	Message Delivery.....	9-5
9.3.3	Message Retention.....	9-5
9.3.4	Routing Support.....	9-5
9.3.4.1	Content-Based Routing.....	9-5
9.3.5	Partitioning	9-5
9.3.6	High Availability	9-6
9.3.7	Backup and Recovery.....	9-7
9.4	Real Application Clusters Configuration	9-7
9.4.1	OracleAS Integration InterConnect Adapters Supporting Real Application Clusters.....	9-8
9.4.1.1	Adapter Failover Mechanism	9-8
9.4.2	Configuration	9-8
9.4.3	Sample Database Adapter adapter.ini File that Shows the Spoke Database Entry ...	9-9

10 Using InterConnect Manager

10.1	Overview of InterConnect Manager	10-1
10.2	Starting InterConnect Manager	10-1
10.3	Using InterConnect Manager	10-2
10.3.1	Hub	10-2
10.3.1.1	Hub Queue Management	10-3
10.3.1.1.1	List Messages	10-3
10.3.1.1.2	Delete Messages.....	10-4
10.3.1.1.3	Export Messages To a File.....	10-5
10.3.1.1.4	Import Messages From a File.....	10-6

10.3.1.2	Error Message Resubmission.....	10-7
10.3.1.2.1	List Messages	10-7
10.3.1.2.2	Resend Messages.....	10-7
10.3.2	Adapters.....	10-8
10.3.2.1	Configuration File Management	10-9
10.3.2.2	Error Management	10-9
10.3.2.2.1	List Rules	10-10
10.3.2.2.2	Add Rule.....	10-10
10.3.2.2.3	View Rule	10-11
10.3.2.2.4	Update Rule.....	10-11
10.3.2.2.5	Delete Rule	10-11
10.3.2.2.6	Set Mail Server	10-11
10.3.2.2.7	View Mail Server	10-11
10.3.3	Repository	10-11
10.3.4	Message Tracking	10-12
10.4	Using InterConnect Manager in Silent Mode	10-13

A Integration Scenario

A.1	Integration Scenario Overview	A-1
A.1.1	The New Centralized System.....	A-1
A.1.2	The Legacy System	A-1
A.1.3	The Integration Scenario	A-2
A.2	Modeling the Integration	A-2
A.3	Implementing the Scenario	A-3
A.3.1	Review Legacy System Database Trigger	A-4
A.3.2	Create a Project.....	A-4
A.3.3	Create the Common View Business Object.....	A-5
A.3.4	Create Business Object Events	A-5
A.3.4.1	DTD Code	A-6
A.3.5	Create Applications	A-7
A.3.6	Create a Cross Reference Table.....	A-8
A.3.7	Create Publish Events	A-8
A.3.8	Subscribe to Events.....	A-11
A.3.8.1	DBAPP Application Subscriptions.....	A-11
A.3.8.2	AQAPP Application Subscriptions.....	A-16
A.3.9	Create Content-based Routing.....	A-17
A.3.10	Create an Oracle Workflow Process Bundle.....	A-18
A.3.11	Deploy the Process Bundle to Oracle Workflow.....	A-19
A.3.12	Creating Objects in Oracle Workflow for Modeling.....	A-20
A.3.12.1	Message.....	A-21
A.3.12.2	Lookup Type	A-21
A.3.12.3	Notification.....	A-21
A.3.12.4	What Oracle Workflow provides.	A-21
A.3.12.5	Copy Lookup Type (Approval).....	A-21
A.3.12.6	Create an Oracle Workflow Message	A-21
A.3.12.7	Create an Oracle Workflow Notification	A-22
A.4	Modeling Business Logic in Oracle Workflow	A-23

A.5	Deployment	A-25
A.5.1	Setting Queues	A-25
A.5.2	Sync Adapters	A-26
A.5.3	Exporting and Installing Code.....	A-27
A.6	Conclusion	A-27

B Using the Data Definition Description Language

B.1	About D3L.....	B-1
B.1.1	What Is D3L?	B-1
B.1.2	When Is D3L Used?	B-2
B.1.3	D3L Features.....	B-2
B.1.3.1	Integrate Transport Properties	B-3
B.1.3.2	Allow Multiple Imparrays	B-4
B.2	Native Format Message and D3L File Example	B-4
B.2.1	Description of Native Format Message Contents in a D3L File.....	B-5
B.2.2	Configuration of Native Format Message with a D3L File.....	B-5
B.2.2.1	adapter.ini Parameter File Setting.....	B-6
B.2.2.2	Message Header Attributes.....	B-6
B.2.2.2.1	Name/Value Pair Message Header Attributes.....	B-6
B.2.2.2.2	Magic Value Message Header Attribute	B-7
B.3	D3L File Structure	B-8
B.3.1	Supported D3L Data Types.....	B-10
B.3.1.1	Signed or Unsigned Integers.....	B-10
B.3.1.2	Floating Point Numbers	B-12
B.3.1.3	Strings.....	B-13
B.3.1.4	Structures	B-17
B.3.1.5	Sequences.....	B-17
B.3.1.6	Data Padding.....	B-20
B.3.2	Comma-Separated Values File Parsing with D3L.....	B-21
B.3.2.1	CSVs are Assigned to Named Fields	B-21
B.3.2.2	All CSVs are Read into an Array	B-22
B.3.2.3	Delimiter Encoding Styles	B-22
B.4	D3L Integration with OracleAS Integration InterConnect Adapters	B-23
B.4.1	Runtime Initialization	B-23
B.4.2	Native Format Message to Common View Incoming Message Translations.....	B-24
B.4.3	Common View to Native Format Message Outgoing Messages Translations	B-25
B.5	Installing D3L	B-26
B.6	Configuring D3L	B-27
B.6.1	Task 1: Configure D3L with iStudio	B-27
B.6.2	Task 2: Create a Native Format Message	B-27
B.6.3	Task 3: Create a D3L File Describing the Native Format Message	B-27
B.6.4	Task 4: Configure a Native Format Message with a D3L File.....	B-28
B.6.5	Task 5: Configure D3L with OracleAS Integration InterConnect Adapters	B-28
B.6.6	Task 6: Import a D3L File in iStudio	B-29
B.6.7	Task 7: Define Metadata Properties with Each Event (Optional).....	B-30
B.7	D3L Use Case.....	B-31
B.7.1	D3L Use Case Overview	B-31

B.7.2	Creating Data Type Definitions for Application Views.....	B-32
B.7.2.1	Task 1: Create a DTD File for the Advanced Queuing Adapter.....	B-32
B.7.2.2	Task 2: Create a D3L File for the FTP Adapter.....	B-32
B.7.3	Configuring the aqapp_pub and fileapp_sub Applications in iStudio	B-34
B.7.3.1	Task 1: Create a New Workspace and New Project	B-34
B.7.3.2	Task 2: Create the Employee Business Object	B-35
B.7.3.3	Task 3: Create the newEmployee Event.....	B-35
B.7.3.4	Task 4: Create the aqapp_pub Application.....	B-36
B.7.3.5	Task 5: Enable the aqapp_pub Application to Publish the newEmployee Event.....	B-36
B.7.3.5.1	Select the Event to Publish.....	B-36
B.7.3.5.2	Define the Application View	B-37
B.7.3.5.3	Define the Application View to Common View Mapping.....	B-38
B.7.3.6	Task 6: Define the Application Queue for the aqapp_pub Application.....	B-39
B.7.3.7	Task 7: Create the fileapp_sub Application.....	B-40
B.7.3.8	Task 8: Enable the fileapp_sub Application to Subscribe to the newEmployee Event	B-40
B.7.3.8.1	Select the Event to which to Subscribe	B-40
B.7.3.8.2	Define the Application View	B-40
B.7.3.8.3	Define the Application View to Common View Mapping.....	B-41
B.7.4	Installing the Advanced Queuing and FTP Adapters.....	B-42
B.7.4.1	Task 1: Install the Advanced Queuing Adapter for Application aqapp_pub ..	B-42
B.7.4.2	Task 2: Create the Application Queue AQAPP_NEWEMP	B-42
B.7.4.3	Task 3: Install the FTP Adapter for Application fileapp_sub	B-43
B.7.4.4	Task 4: Copy the newemp.xml D3L File to the fileapp_sub Adapter Directory	B-44
B.7.4.5	Task 5: Set the D3L file and Payload Type in the adapter.ini Adapter Initialization File	B-44
B.7.5	Running the D3L Use Case.....	B-44
B.7.5.1	Task 1: Start the Adapters	B-44
B.7.5.1.1	To Start the Adapters on UNIX:.....	B-44
B.7.5.1.2	To Start the Adapters on Windows:	B-45
B.7.5.2	Task 2: Create PL/SQL Code to Trigger the Native newEmployee Event.....	B-45
B.7.5.3	Task 3: Trigger the newEmployee Event	B-46
B.7.5.4	Task 4: Verify Receipt of newEmployee Event	B-47
B.7.6	Using Other Adapters in D3L and XML Modes	B-47
B.7.6.1	Using the HTTP, SMTP, or MQ Series Adapters in D3L Mode.....	B-48
B.7.6.2	Using XML Mode	B-48
B.8	Additional D3L Sample Files and DTD.....	B-48
B.8.1	Additional D3L Sample Files	B-49
B.8.1.1	Sample File with Structure VehicleRegistration	B-49
B.8.1.2	Sample File with Structure Hierarchy PersonRecord	B-50
B.8.1.3	Sample File with Structure ProductRecord	B-51
B.8.2	D3L DTD	B-52

C Transformations

C.1	OracleAS Integration InterConnect Transformations	C-1
-----	---	-----

C.1.1	Copy Fields	C-1
C.1.2	Copy Object	C-1
C.1.3	Concat Fields	C-1
C.1.4	Expand Fields	C-1
C.1.5	Set Constant	C-2
C.1.6	True Conditional Lookup XRef	C-2
C.1.7	True Conditional Lookup DVM	C-2
C.1.8	Conditional Copy.....	C-2
C.1.9	True Conditional Copy	C-3
C.1.10	True Conditional Concat	C-3
C.1.11	True Conditional To Number	C-3
C.1.12	False Conditional Copy.....	C-3
C.1.13	False Conditional Concat.....	C-3
C.1.14	False Conditional To Number.....	C-4
C.1.15	To Number.....	C-4
C.1.16	Substring	C-4
C.1.17	Char Replace.....	C-4
C.1.18	String Replace.....	C-5
C.1.19	LTrim	C-5
C.1.20	RTrim	C-5
C.1.21	LPad	C-5
C.1.22	RPad.....	C-6
C.1.23	Lookup XRef.....	C-6
C.1.24	Delete XRef	C-6
C.1.25	Lookup DVM.....	C-6
C.1.26	Truncate.....	C-7
C.1.27	Increment	C-7
C.1.28	DatabaseOperation	C-7

D Troubleshooting OracleAS Integration InterConnect

D.1	Problems and Solutions	D-1
D.1.1	iStudio Fails to Connect	D-1
D.1.2	Mappings in iStudio	D-2
D.1.3	Metadata Not Editable in iStudio.....	D-3
D.1.4	Subscribing Adapter Does Not Receive Messages From the Hub	D-3
D.1.5	Messages Are Not Getting Delivered to a Spoke Application.....	D-3
D.1.6	OracleAS Integration InterConnect Repository Does Not Start	D-4
D.1.7	DB Adapter Does Not Pick Up Messages That Have Been Published From the Database D-5	
D.1.8	FTP Adapter Cannot Match Incoming Message With Any D3L Definiton	D-5
D.1.9	AQ Adapter Does Not Pick Up Mesage From the Spoke AQ.....	D-5
D.1.10	CBR Issue with DB Adapter.....	D-6
D.2	Need More Help?.....	D-6

Glossary

Index

Send Us Your Comments

Oracle Application Server Integration InterConnect User's Guide, 10g Release 2 (10.1.2)

Part No. B14069-01

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

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

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

- Electronic mail: appserverdocs_us@oracle.com
- FAX: 650-506-7356 Attn: Oracle Application Server Documentation Manager
- Postal service:

Oracle Corporation
Oracle Application Server Documentation Manager
500 Oracle Parkway, M/S 10p6
Redwood Shores, CA 94065
USA

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

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

Preface

This Preface contains these topics:

- [Audience](#)
- [Documentation Accessibility](#)
- [Structure](#)
- [Related Documents](#)
- [Conventions](#)

Audience

This guide is targeted at the following types of users:

- Business analysts and integration engineers, for iStudio.
- System Administrators, for the runtime component.

The audience should have the following prerequisites, which are discussed but not explained:

- Domain knowledge of the applications being integrated.
- Database concepts and working knowledge of SQL, PL/SQL, or SQL* Plus.

Documentation Accessibility

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

<http://www.oracle.com/accessibility/>

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Structure

This document contains:

Chapter 1, "Getting Started with OracleAS Integration InterConnect"

Introduces OracleAS Integration InterConnect and presents an overview of the product and the tools.

Chapter 2, "Using iStudio"

Describes iStudio and how to create workspaces and projects.

Chapter 3, "Creating Applications, Common Views, and Business Objects"

Describes how to create and manage applications, common views, and business objects using iStudio.

Chapter 4, "Using Events in iStudio"

Describes using iStudio to create, publish, and subscribe to events.

Chapter 5, "Using Procedures in iStudio"

Describes using iStudio to create, invoke, and implement procedures.

Chapter 6, "Enabling Infrastructure"

Describes the enabling infrastructure tasks in iStudio including creating cross reference tables and domain value mappings.

Chapter 7, "Using Oracle Workflow"

Describes how OracleAS Integration InterConnect works with Oracle Workflow.

Chapter 8, "Deployment"

Describes the deployment tasks in iStudio.

Chapter 9, "Runtime System Concepts and Components"

Describes the runtime components and concepts of OracleAS Integration InterConnect.

Chapter 10, "Using InterConnect Manager"

Introduces the InterConnect Manager and describes how you use it to manage your integration environment.

Appendix A, "Integration Scenario"

Provides an integration scenario and model based on a fictitious company, Acme, Inc. using OracleAS Integration InterConnect.

[Appendix B, "Using the Data Definition Description Language"](#)

Describes how to use the data definition description language (D3L) in native format message-to-application view and application view-to-native format message translations.

[Appendix C, "Transformations"](#)

Provides a list of OracleAS Integration InterConnect transformations.

[Appendix D, "Troubleshooting OracleAS Integration InterConnect"](#)

Describes common problems that you might encounter when using OracleAS Integration InterConnect and explains how to solve them.

Related Documents

For more information, refer to these Oracle resources:

- [Oracle Application Server Integration InterConnect Installation Guide](#)
- [Oracle Application Server Integration InterConnect Release Notes](#)

Printed documentation is available for sale in the Oracle Store at

<http://oraclestore.oracle.com/>

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

<http://www.oracle.com/technology/membership/>

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

<http://www.oracle.com/technology/documentation/>

Conventions

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

- [Conventions in Text](#)
- [Conventions in Code Examples](#)
- [Conventions for Windows Operating Systems](#)

Conventions in Text

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

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an index-organized table .

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

Conventions in Code Examples

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

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

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

Convention	Meaning	Example
[]	Anything enclosed in brackets is optional.	DECIMAL (<i>digits</i> [, <i>precision</i>])
{ }	Braces are used for grouping items.	{ENABLE DISABLE}
	A vertical bar represents a choice of two options.	{ENABLE DISABLE} [COMPRESS NOCOMPRESS]
...	Ellipsis points mean repetition in syntax descriptions. In addition, ellipsis points can mean an omission in code examples or text.	CREATE TABLE ... AS <i>subquery</i> ; SELECT <i>col1</i> , <i>col2</i> , ... , <i>coln</i> FROM employees;
Other symbols	You must use symbols other than brackets ([]), braces ({ }), vertical bars (), and ellipsis points (...) exactly as shown.	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;

Convention	Meaning	Example
<i>Italics</i>	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/ <i>system_password</i> DB_NAME = <i>database_name</i>
UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. Because these terms are not case sensitive, you can use them in either UPPERCASE or lowercase.	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
lowercase	Lowercase typeface indicates user-defined programmatic elements, such as names of tables, columns, or files. Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;

Conventions for Windows Operating Systems

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

Convention	Meaning	Example
Click Start , and then choose the <i>menu item</i>	How to start a program.	To start the Database Configuration Assistant, click Start , and choose Programs . In the Programs menu, choose Oracle - HOME_NAME and then click Configuration and Migration Tools . Choose Database Configuration Assistant .
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the filename begins with \\, then Windows assumes it uses the Universal Naming Convention.	c:\winnt"\"system32 is the same as C:\WINNT\SYSTEM32
C:\>	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the <i>command prompt</i> in this manual.	C:\oracle\oradata>
Special characters	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	C:\>exp HR/HR TABLES=employees QUERY=\"WHERE job_id='SA_REP' and salary<8000\"

Convention	Meaning	Example
<i>HOME_NAME</i>	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	C:\> net start Oracle <i>HOME_NAME</i> TNSListener
<i>ORACLE_HOME</i> and <i>ORACLE_BASE</i>	<p>In releases prior to Oracle8i release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level <i>ORACLE_HOME</i> directory.</p> <p>This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level <i>ORACLE_HOME</i> directory. There is a top level directory called <i>ORACLE_BASE</i> that by default is C:\oracle\product\10.1.0. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is C:\oracle\product\10.1.0\db_<i>n</i>, where <i>n</i> is the latest Oracle home number. The Oracle home directory is located directly under <i>ORACLE_BASE</i>.</p> <p>All directory path examples in this guide follow OFA conventions.</p> <p>Refer to <i>Oracle Database Installation Guide for Windows</i> for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.</p>	Change to the <i>ORACLE_BASE</i> \ <i>ORACLE_HOME</i> \rdbms\admin directory.

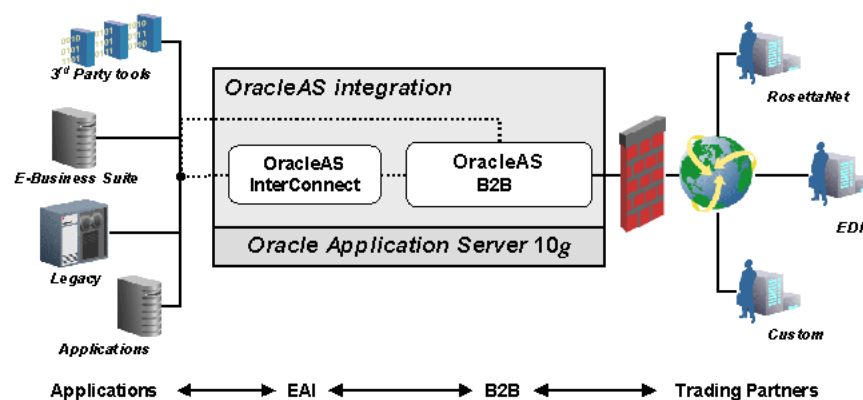
Getting Started with OracleAS Integration InterConnect

This chapter provides an overview of Oracle Application Server Integration InterConnect (OracleAS Integration InterConnect), its features, and components. It contains the following topics:

- [What is OracleAS Integration InterConnect?](#)
- [Standard Messaging](#)
- [OracleAS Integration InterConnect Integration Process](#)
- [What's New in This Release?](#)

1.1 What is OracleAS Integration InterConnect?

OracleAS Integration InterConnect is an integral component of Oracle Application Server and provides a comprehensive application integration framework to enable seamless integration of enterprise software. It is built on top of the Oracle Application Server integration platform and leverages its underlying services. It is designed to integrate heterogeneous systems, such as Oracle, non-Oracle, and legacy applications. Together with OracleAS Integration B2B, it provides a complete end-to-end for integrating your enterprise and beyond. OracleAS Integration B2B provides extensive protocol support to enable the deployment of industry-recognized Business-to-Business (B2B) standards: RosettaNet, Electronic Data Interchange (EDI), Applicability Statement 2 (AS2), and custom configurations.



OracleAS Integration InterConnect provides the following value proposition:

- **Speed of Integration Development:** Elevate the integration problem from a technical coding exercise to a functional modeling exercise, thereby reducing or eliminating the programming effort normally associated with integration. This ensures that the development time is reduced significantly.
- **Speed of Runtime Integration Execution:** Minimize latency and maximize throughput for real-time cross-application integration. This ensures that the integration solution is performant.
- **Speed of Integration Evolution:** Expose an integration methodology that promotes reuse of existing integration logic and minimizes change impact. As the integration scenario evolves over time (existing applications are upgraded, new applications are added, old applications are removed), the change impact is limited to just the application that is undergoing the change. The other applications are shielded from these changes. This reduces the complexity and management issues that arise over the integration lifecycle.

1.1.1 OracleAS Integration InterConnect Components

OracleAS Integration InterConnect has the following core components:

- [OracleAS Integration InterConnect Hub](#)
- [OracleAS Integration InterConnect Adapters](#)
- [OracleAS Integration InterConnect Development Kit](#)

1.1.1.1 OracleAS Integration InterConnect Hub

The hub consists of a middle tier repository server program communicating with a database. The repository has the following functionality:

- At design time, all integration logic defined in iStudio is stored in tables in the repository as metadata.
- At runtime, the repository provides access to this metadata for adapters to integrate applications.

The repository server is deployed as a standalone Java application running outside the database. The repository schema is a set of tables in the Oracle Application Server hub database.

Note: From the current release of OracleAS Integration InterConnect, the adapters and iStudio connect to the repository server using RMI (Remote Method Invocation) instead of CORBA (Common Object Request Broker Architecture).

1.1.1.2 OracleAS Integration InterConnect Adapters

Adapters perform two major tasks:

- Provide connectivity between an application and the hub.
- Transform and route messages between the application and the hub.

Adapters are deployed as standalone Java applications running outside the database. Adapters can be deployed in the following configurations:

- Co-located with the OracleAS Integration InterConnect Hub

- Co-located with the application they are connecting to
- Located on a separate computer altogether

See Also: ["Using Adapters for Integration"](#) on page 1-8

1.1.1.3 OracleAS Integration InterConnect Development Kit

iStudio is a design time integration specification tool targeted at business analysts. This tool helps business analysts specify the integration logic at a functional level, instead of a technical coding level. iStudio exposes the integration methodology using simple wizards and reduces, or eliminates, the need for writing code to specify the integration logic. This reduces the total time required to complete an integration.

iStudio is a multiuser tool with fine-grained locking for all OracleAS Integration InterConnect first class objects. As a result, multiple users can work simultaneously on the same integration scenario without compromising the consistency of the metadata.

iStudio allows business analysts to complete the following tasks:

- Define data to be exchanged across applications.
- Semantically map data across applications.
- Define the business process collaboration across applications using Oracle Workflow and associate the semantic maps with business processes, if required.
- Configure and deploy the integration.

iStudio is deployed as a standalone Java application running outside the database. It can be deployed on any computer with access to the hub computer running Windows.

See Also: [Chapter 2, "Using iStudio"](#)

1.1.1.3.1 OracleAS Integration InterConnect SDKs OracleAS Integration InterConnect Software Development Kit (SDK) allows you to customize OracleAS Integration InterConnect to meet your integration needs.

iStudio SDK The iStudio SDK is a collection of Java `jar` and `Javadoc` files usually deployed on the same computer as iStudio. The iStudio SDK is only available on Windows. Using the iStudio SDK and Java, users can build the following:

- New transformation functions
- New browsers to import application-native data structures and APIs into iStudio

Documentation and samples are provided with the iStudio SDK.

Adapter SDK The Adapter SDK is a collection of Java `jar` and `Javadoc` files that can be deployed on any computer. The Adapter SDK is available on all tier one platforms. Using the Adapter SDK, users can write new adapters in Java for applications or protocols not currently supported by OracleAS Integration InterConnect. Specifically, only the bridge subcomponent must be written. The agent is a generic engine already written and is part of each adapter.

Documentation and samples are provided with the Adapter SDK.

Oracle Workflow Oracle Workflow provides a comprehensive business process management system that enables traditional workflow applications, and process collaboration in a single solution. Using Oracle Workflow Business Event System, OracleAS Integration InterConnect can model an integration solution on business processes. With OracleAS Integration InterConnect and Oracle Workflow, business

collaborations across two or more applications can be defined to implement the organization's business processes.

1.2 Standard Messaging

OracleAS Integration InterConnect provides the following basic services expected of a messaging middleware platform:

- **Guaranteed delivery of messages:** All messages have guaranteed delivery end-to-end. Messages are delivered exactly once and in the order sent.
- **Scalability:** Multiple adapters are instantiated to serve one application. The hub runs in an Oracle Real Application Clusters environment.
- **Load Balancing:** Messages can be partitioned based on load between multiple adapters servicing one application. One or more adapters can serve all messages for one application. In addition, one or more adapters can be dedicated per integration point in which the application participates.
- **Runtime Management:** The Oracle InterConnect Manager helps manage the integration scenario and components at runtime. The IC Manager allows users to start and stop components, monitor message flow, detect problems, and manage errors.
- **Deployment Support:** The messaging hub consists of Advanced Queues that are configured for runtime. You can configure the number of queues to create, name these queues, and match adapters with messages in a specific named queue.

The following supplementary features do not require any additional coding:

- **Content-based Routing:** Route messages by building business rules based on message content. For example, a procurement system routes fulfillment requests to different fulfillment centers based on an originating location.
- **Cross-referencing:** Correlate keys that uniquely identify the entities in one application with corresponding entities created in other applications. For example, a purchase order created in a procurement system has a native ID *X*. The purchase order is then routed to a fulfillment system, where it is created with native ID *Y*. As a result, *X* and *Y* must be cross referenced for OracleAS Integration InterConnect to correlate communication about this same logical entity in two different systems without each system understanding the native ID of the other system.
- **Domain Value Mapping:** Map code tables across systems. For example, a purchase order in a procurement system has a PO Status field with domain values, `Booked` and `Shipped`. The corresponding field in a fulfillment system has the domain values, `1` and `2`. OracleAS Integration InterConnect allows the user to create the mappings `booked=1`, `shipped=2` so that it can correlate these values at runtime without each system understanding the domain value set of the other system.

1.2.1 Supported Messaging Paradigms

OracleAS Integration InterConnect supports the following messaging paradigms. These paradigms are defined in iStudio at design time. The definitions are used at runtime to route the messages suitably:

- **Publish/Subscribe Messaging:** An application publishes a message if it sends data out to the OracleAS Integration InterConnect hub without knowing the destination applications. In addition, data is not expected in return. An application subscribes to a message if it receives the data from the OracleAS Integration

InterConnect hub regardless of the application that sent the data. Also, it does not send any data out in return. Events in iStudio are used to model this paradigm.

- Request/Reply Messaging: An application publishes a message and expects a message in return as a reply. The application subscribing to the request sends a reply back to the sender after processing the request. Procedures in iStudio are used to model this paradigm. Request/Reply has two types of messaging:
 - Synchronous: The application making the request is blocked until it receives a reply.
 - Asynchronous: The application makes the request and proceeds with normal processing. It does not wait for a response. A reply is delivered asynchronously and is consumed by the application.
- Point-to-Point Messaging: Both Publish/Subscribe and Request/Reply can acquire a point-to-point characteristic if the sending application explicitly specifies which application should receive the message. This can be modeled using content-based routing in iStudio.

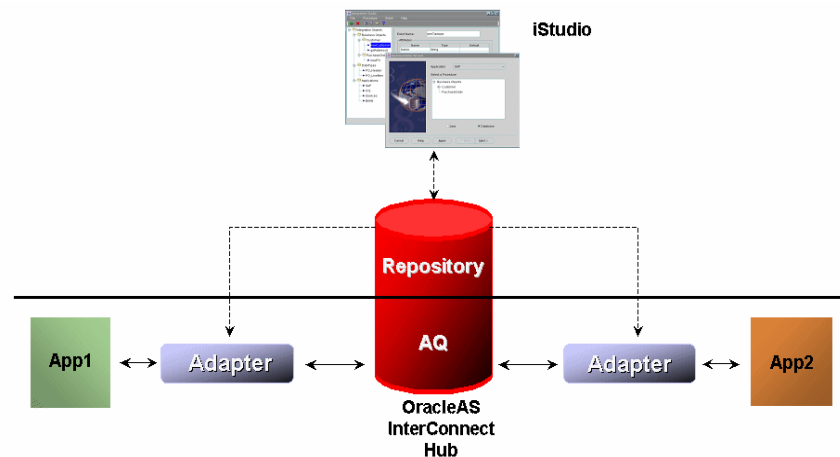
1.3 OracleAS Integration InterConnect Integration Process

Application integration using OracleAS Integration InterConnect involves the following two phases:

- Design Time
- Runtime

Figure 1–1 provides an overview of design time and runtime phases in integration.

Figure 1–1 A Graphical Overview of Design Time and Runtime Phases in Integration



1.3.1 Design Time

During the design time, a business analyst uses iStudio to define the integration objects, applications that participate in the integration, and the specifications of the data exchanged between applications. All the specifications are stored as metadata in the OracleAS Integration InterConnect Repository.

1.3.2 Runtime

One or more OracleAS Integration InterConnect adapters are configured to service each application participating in the integration. At runtime, if an application is sending messages out, the adapters attached to it will retrieve the metadata from the repository to receive messages from the application, determine their formats, perform transformations, and route to corresponding queues in the OracleAS Integration InterConnect hub. For applications receiving messages, the adapters attached to it will retrieve the metadata from the repository to receive messages from the OracleAS Integration InterConnect hub queues, determine their formats, perform transformations and then deliver the messages to the application.

1.3.3 Separation of Integration Logic and Platform Functionality

Integration using OracleAS Integration InterConnect is a two-step process. During design time, integration logic is modeled in iStudio and captured in the repository as metadata. Metadata is created in the repository using iStudio during design time and is represented by application views, common views, and transformations. At runtime, the underlying services treat this metadata as runtime instructions to enable the conversation among participating applications. Integration has two components:

- **Integration logic:** Consists of the business rules and transformation logic necessary to integrate heterogeneous systems. Using iStudio, this integration logic can be modeled and the results stored in the repository as metadata.
- **Platform functionality:** Consists of the integration infrastructure provided with OracleAS Integration InterConnect and the Oracle database. In addition, OracleAS Integration InterConnect provides application and protocol adapters. The platform services provide the requisite infrastructure necessary for integration.

1.3.4 Unique Integration Methodology

iStudio exposes an integration methodology that eliminates the complexities of point-to-point custom integration solutions. The integration methodology is based on a hub-and-spoke model.

1.3.4.1 How the Hub-and-Spoke Methodology Works

An integration point is the context, which ties in a particular message exchange, between two or more participating applications in the integration scenario. OracleAS Integration InterConnect supports two types of integration points:

- **Events:** This type of integration point is used to model the publish/subscribe messaging paradigm.

Create Customer: For example, an integration scenario may require that customer information across two applications be synchronized in real time. Whenever a new customer is created in the application, `App1`, the customer should also be created in the application, `App2`. `Create_Customer` is an *event* that triggers the communication between the two applications. `App1` produces the information, and `App2` consumes it.

- **Procedures:** This type of integration point is used to model the request/reply messaging paradigm.

Get Item Info: For example, a user of `App1` may request information about an item stored in `App1`. The information about that item might be segmented across the two applications. To give a meaningful response to the user of `App1`, it is necessary to query `App2` for information about the item. `Get_Item_Info` is an

integration point between the two applications because it triggers communication between the two applications. App1 produces a query and App2 consumes it. App2 produces the response and App1 consumes it.

The common view consists of a list of such integration points, each with its own associated data. Applications participate in the integration by binding to one or more of these common view integration points.

For each binding, applications have their own application view of data that needs to be exchanged. Each binding involves a mapping, or transformation, between the application view and the common view in the context of the integration point. In this model, the application views are the spokes and the common view is the hub.

Create_Customer is an integration point. If the information to exchange is only the new customer's name, the common view has all the information potentially captured in a name defined in an application-independent method. This information must be a superset of all the information that needs to be exchanged across App1 and App2.

Prefix, First Name, Last Name, Middle Initial, Maiden Name, Suffix is an example of a common view customer name definition.

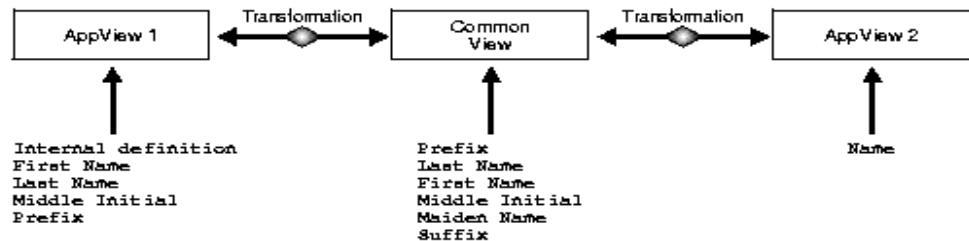
Now, App1's internal *definition* of name (App1's application view) could be First Name, Last Name, Middle Initial, Prefix.

The application view for App2 could be Name (one field that describes Last Name, First Name).

When App1 sends this information out or publishes an event, transformations are defined from its application view to the common view. When App2 receives this information or subscribes to an event, transformations are defined from the common view to its application view.

Figure 1–2 illustrates this example within the hub-and-spoke model where the common view is the hub, and the application views are the spokes.

Figure 1–2 OracleAS Integration InterConnect Hub-and-Spoke Model



The hub-and-spoke model has the following advantages:

- **Loosely Coupled Integration:** If applications are integrated directly with each other, then any change in one application will result in changes required for the other applications. In OracleAS Integration InterConnect, applications integrate to the common view and not directly with each other. This reduces the number of integration interfaces.
- **Easy Customization:** If an application is upgraded or changed, then only the corresponding application view needs to be remapped to the hub. The other spokes and their relationships with the hub remain unchanged. This localizes the change impact to the affected application.

- **Easy Extensibility:** If an application is added or removed from the integration scenario, then other integrated applications are not affected. For example, if a new application is added to the integration scenario, it must define its spoke component (the application view) and map that component to the hub (common view) on a per integration point basis. This does not affect other applications in the integration.
- **Enhanced Reusability:** If the common view of an application is already built, then this common view can be reused to integrate the application with any other applications. For example, to integrate the Marketing CRM module to SAP, the integration would be from iMarketing to common view to SAP. If there is a requirement to integrate iMarketing to Peoplesoft, then the iMarketing to common view integration can be reused. Only the common view to the Peoplesoft integration needs to be built.

1.3.5 Integration Lifecycle Management

Managing, customizing, and evolving an integration over time is as important as creating the integration in the first place. The hub-and-spoke integration model has advantages to help achieve this goal. In addition, the OracleAS Integration InterConnect repository, which contains all the integration logic, provides extensive services for managing changes over time. The repository provides fine-grained versioning of all OracleAS Integration InterConnect first class objects such as events, messages, and data types. Some important aspects of versioning to aid the lifecycle support include:

- **Basic Versioning:** New versions of first class objects, such as messages, can be created to address changing integration needs. Different versions of the same object can co-exist in the repository. This approach has two advantages:
 - Eliminates the need for an expanded namespace to address modifications.
 - Allows related entities to be grouped together for easy management.
- **Multiple Active Versions:** Multiple versions of the same message can be active in the same integration scenario simultaneously. This helps transition and integration incrementally without requiring changes to existing messages. For example, if a purchase order definition for an application or the application view of the purchase order needs to change, a new version of the message can be created and activated for that application. Once this metadata is created, the application can smoothly transition from sending and receiving messages based on the old definitions to the new one.
- **Migration Support:** Different versions of metadata can be migrated across repositories on a first class object basis. This feature allows fine-grained control of content in different repositories, such as a development repository and a production repository.
- **Consistency Control:** OracleAS Integration InterConnect detects and flags metadata conflicts. This helps prevent accidental overwriting of metadata and maintains consistency of metadata in the repository.

1.3.6 Using Adapters for Integration

Adapters are runtime components, which process integration logic captured in the repository as runtime instructions, to enable the integration. Prepackaged adapters help applications at runtime to participate in the integration without any programming effort.

Adapters perform the following tasks:

- **Application Connectivity:** Adapters connect applications with OracleAS Integration InterConnect hub to transfer data between them. The logical subcomponent within an adapter that handles this responsibility is called a bridge. This is the protocol/application-specific piece of the adapter that communicates with the application.

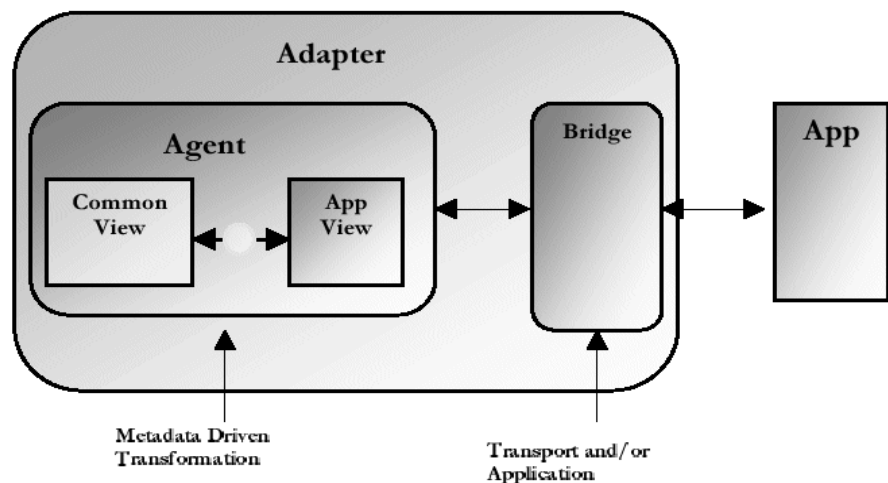
For example, the database adapter can connect to an Oracle database using JDBC and runtime SQL APIs. The bridge subcomponent only knows how to call the correct APIs.

- **Transformations:** Transform data from the application view to common view and conversely as dictated by the repository metadata. In general, adapters are responsible for carrying out all the runtime instructions captured through iStudio as metadata in the repository. Transformations are an important subset of these instructions. The logical subcomponent within an adapter that handles the runtime instructions is called an agent. This is the generic runtime engine in the adapter that is independent of the application to which the adapter connects. It focuses on the integration scenario based on the integration metadata in the repository. There is no integration logic coded into the adapter itself. All integration logic is stored in the repository. The repository contains the metadata that drives this subcomponent.

In the preceding database adapter example, the bridge subcomponent knows which SQL APIs to call, but not how to call them. All adapters have the same agent code but the metadata is different. This difference in metadata controls and differentiates the behavior of each adapter.

The OracleAS Integration InterConnect Adapter Architecture is displayed in [Figure 1-3](#).

Figure 1-3 OracleAS Integration InterConnect Adapter Architecture



See Also: *OracleAS Integration InterConnect Installation Guide* for a complete list of OracleAS Integration InterConnect Adapters

1.4 What's New in This Release?

Oracle Application Server Integration InterConnect 10g Release 2 (10.1.2) introduces a number of new features to provide simplified configuration and management.

Recursive DTD support

OracleAS Integration InterConnect 10g Release 2 (10.1.2) is designed to handle recursion in data types, with support in iStudio and all adapters.

RMI Implementation

The current release of OracleAS Integration InterConnect implements Remote Method Invocation (RMI) as the communication protocol for distributed computing, replacing the existing CORBA implementation.

HTTP Adapter Synchronous Request/Reply support

In this release of OracleAS Integration InterConnect, the HTTP adapter supports the synchronous request/reply scenario, in addition to the publish/subscribe model. The functionality will enable the HTTP adapter to send the status codes and also enable synchronous replies for its requests.

IC Manager

InterConnect Manager is a new utility that takes care of both the runtime management and error handling needs of OracleAS Integration InterConnect. Oracle Enterprise Manager is no longer required for managing OracleAS Integration InterConnect.

Complete HA support

OracleAS Integration InterConnect uses Oracle Process Manager and Notification (OPMN), Oracle Database Server, and Oracle Real Application Clusters to enable high availability for its components. OracleAS Integration InterConnect ensures a complete high availability support at three different levels: process-level, node-level, and site-level.

Enhancements in iStudio

In this release of OracleAS Integration InterConnect, a number of enhancements have been made to iStudio, such as the deployment of generated procedures through iStudio, and enhancement in iStudio usability.

Oracle Applications Adapter

A new adapter called the Oracle Applications adapter, designed specifically to support Oracle Applications, has been included in this release.

Using iStudio

This chapter describes iStudio and its concepts. It contains the following topics:

- [Overview of iStudio](#)
- [Starting iStudio](#)
- [Parts of the iStudio Window](#)
- [Using Workspaces in iStudio](#)
- [Using Projects in iStudio](#)

2.1 Overview of iStudio

iStudio is a design time integration specification tool used to help business analysts specify the integration logic at a functional level, instead of a technical coding level. iStudio exposes the integration methodology using simple wizards and reduces or eliminates the need for writing code to specify the integration logic. This reduces the total time required to complete an integration.

iStudio is a multiuser tool with fine-grained locking for all OracleAS Integration InterConnect first-class objects. This allows multiple users to work simultaneously on the same integration scenario without compromising the consistency of the metadata.

iStudio allows business analysts to perform the following tasks:

- Define the data that needs to be exchanged across applications.
- Semantically map the data across applications.
- Define the business process collaboration across applications and associate the semantic maps with business processes if required.
- Configure and deploy the integration.

iStudio is deployed as a standalone Java application running outside the database. iStudio runs only on Windows and can be deployed on any computer that has access to the hub computer.

See Also: *Oracle Application Server Integration InterConnect Installation Guide*

2.1.1 iStudio Concepts

The following concepts are described:

- [Applications](#)
- [Common Views and Business Objects](#)

- [Transformations or Mappings](#)
- [Metadata Versioning](#)
- [Tracking Fields](#)
- [Content-Based Routing](#)
- [Cross Reference Tables](#)
- [Domain Value Mapping](#)
- [Routing and the Message Capability Matrix](#)

2.1.1.1 Applications

Each component integrated with OracleAS Integration InterConnect is referred to as an application. Each application expresses interest in specific messages, what its internal data type is, and how the message should be mapped to or from that internal type to the external world.

See Also: [Chapter 3, "Creating Applications, Common Views, and Business Objects"](#)

2.1.1.2 Common Views and Business Objects

OracleAS Integration InterConnect follows a hub-and-spoke integration methodology. The common view is the *hub view* of the integration in which each spoke is an application participating in the integration. The common view consists of the following elements:

- **Business Objects:** A collection of logically related integration points. For example, Create Customer, Update Customer, Delete Customer, and Get Customer Info are integration points that logically belong under a Customer business object.
- **Events:** An integration point used to model the Publish/Subscribe paradigm. An event has associated data which is the common view of all the data to be exchanged through this event.
- **Procedures:** An integration point used to model the Request/Reply paradigm. This is a modeling paradigm only, no actual procedures are called. Like events, procedures have associated data that represents the common view of data exchanged through the procedure.
- **Common Data Types:** A data type used to define data for reuse. It's useful to define complex hierarchical data.

See Also: [Chapter 3, "Creating Applications, Common Views, and Business Objects"](#)

2.1.1.2.1 Events An event is an integration point used to model the Publish/Subscribe paradigm. An event is only associated with one data set. An event has associated data that is the common view of all data to be exchanged through the event. The data associated with an event in the common view must be a superset of the data of participating applications. The publish/subscribe paradigm is used for asynchronous one-way communication. The sending application is said to publish the event. The receiving application subscribes to the event.

See Also: [Chapter 4, "Using Events in iStudio"](#)

2.1.1.2.2 Procedures A procedure is an integration point used to model the Request/Reply paradigm. This is a modeling paradigm only, no actual procedures are called. The request/reply paradigm is used for two-way context sensitive communication. This communication can be either synchronous (the requesting application is blocking until it receives a reply) or asynchronous (the requesting application gets the reply asynchronously, it does not block-wait for the response after sending the request). An application can either invoke a procedure to model sending a request and receiving a reply, or implement a procedure to model receiving a request and sending a reply. Similar to events, a procedure has associated data. A procedure has two data sets: one for the request or IN data and one for the reply or OUT data.

See Also: [Chapter 5, "Using Procedures in iStudio"](#)

2.1.1.3 Transformations or Mappings

Transformations are used to map application views of data to their corresponding common views and vice versa. This is used in the context of publishing/subscribing to an event or invoking/implementing a procedure. There are several built-in transformation routines provided with OracleAS Integration InterConnect that are used to build complex mappings. In addition, using the iStudio SDK allows new transformation routines to be created using Java. These transformations can be imported into iStudio and then used similar to a built-in routine.

For example, assume there are two applications, App1 and App2.

App1 is publishing the event and its application view contains the following fields:

First Name

Last Name

Middle Initial

App2 is subscribing to the event and its application view contains the following fields:

Name: One field in the form of LastName, FirstName

When publishing or subscribing to the event, the application view for App1 and App2 must be mapped to the common view using transformations. Assume that the common view event contains the following fields:

Prefix

First Name

Last Name

Middle Initial

Suffix

Thus, there are two transformations performed. In the first, the data from App1 is transformed to the common view, and in the second, the data from the common view is transformed to the application view of App2.

See Also:

- [Chapter 5, "Using Procedures in iStudio"](#)
- [Appendix C, "Transformations"](#)

2.1.1.4 Metadata Versioning

iStudio supports versioning for application and common data types, events, procedures, and messages.

An owner is the creator of the object and only the owner can modify the object. However, other users can create new versions or copy the original object under a new name. The owner is specified when the repository is installed.

The following functionality is available for versioning:

- **Automatic Versioning:** Whenever a new version is created, it automatically gets a version number. For example, assume that an event called `NewCustomerEvent` is created. When this object is created for the first time, the assigned owner is `OAI` and the version is `V1`. The event name is `NewCustomerEvent/OAI/V1`.
- **Modify Object:** The owner is the only user who can modify the contents of an event and the data associated with it. However, the owner cannot change the version number or the name of the event.
- **Create New Version:** If the owner wants to retain the original event but wants to create a new version of the information with modified data, the owner can create a new version. For example, assume `OAI` wants to retain the original version of the `NewCustomerEvent` object but wants a new version of the object with modified data. Using the Create New Version functionality will provide two objects, `NewCustomerEvent/OAI/V1` and `NewCustomerEvent/OAI/V2`.
- **Load Version:** Not all versions of objects are loaded into iStudio. To work with a specific version of an object, use the Load Version capability. When a new version is created, it becomes the current version.
- **Copy Object:** To create a new object that has many elements identical to an already existing object, first load the existing object and copy it in iStudio. Copying the object allows only modifications to the data, and modifications to the name of the event. Once the name has been modified, the existing object and the object with the new name will coexist in the repository.

Note: Names of events must be unique.

See Also: [Chapter 4, "Using Events in iStudio"](#)

2.1.1.5 Tracking Fields

Tracking fields are one or more application view fields in the context of a particular message. If specified in iStudio, tracking fields can be used to track messages at runtime using the Oracle InterConnect Manager. Tracking is performed only from the perspective of the sending application.

For example, if `App1` publishes a new purchase order and specifies the `PO_order` number field as the tracking field, then the user can log in to the runtime console and specify the message to track, or `New Purchase Order` in this case. The user is then prompted to enter the purchase order number to display the corresponding tracking information.

2.1.1.6 Content-Based Routing

Content-based routing allows you to define rules to route messages based on message content. For example, a sales lead generation system can route leads to different sales force automation systems based on the location of the potential customer.

See Also: [Chapter 6, "Enabling Infrastructure"](#)

2.1.1.7 Cross Reference Tables

Keys for corresponding entities created in different applications can be correlated through cross referencing in iStudio.

See Also: [Chapter 6, "Enabling Infrastructure"](#)

2.1.1.8 Domain Value Mapping

Code tables can be mapped across systems using domain value mapping in iStudio.

See Also: [Chapter 6, "Enabling Infrastructure"](#)

2.1.1.9 Routing and the Message Capability Matrix

In the OracleAS Integration InterConnect hub, Advanced Queues in the database are used to store, route, and forward messages from the sending application adapters to the receiving application adapters. The sending adapters evaluate the recipients based on the metadata. The following method is used to route messages.

- Every adapter has one or more queues where it receives messages.
- The Message Capability Matrix allows queues to be specified for receiving messages on a per message per receiving application basis.

Note: By default, OracleAS Integration InterConnect comes preconfigured with one queue named the `oai_hub_queue`. This queue is used for all messages and applications. This queue does not need to be changed unless the single queue implementation causes a performance bottleneck.

2.2 Starting iStudio

The database and repository must be running before logging in to iStudio. To log on to iStudio, perform the following steps:

1. From the Windows Start menu, select **OracleAS Integration InterConnect**.
2. Select **iStudio**.
3. When iStudio starts, the last opened project is automatically loaded into the default workspace.

See Also: ["Creating a New Project"](#) on page 2-11

2.3 Parts of the iStudio Window

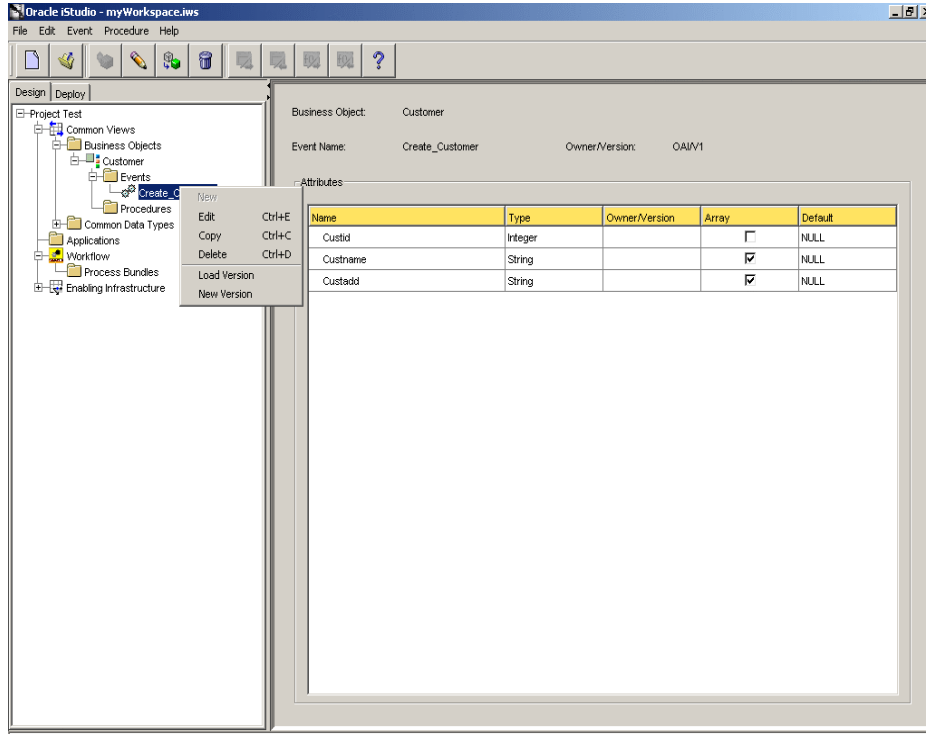
The main iStudio window has the following parts:

- [Menu Bar](#)
- [Toolbar](#)
- [Design Navigation List](#)
- [Deploy Navigation List](#)
- [Context Menus](#)

- [Detail View](#)

When iStudio is started, the main window is displayed as shown in [Figure 2–1](#).

Figure 2–1 OracleAS Integration InterConnect iStudio



2.3.1 Menu Bar

The menu bar provides access to all commands. Click each menu to display its commands. Click a command to run it. There are five menus:

- File Menu
- Edit Menu
- Procedure Menu
- Event Menu
- Help Menu

2.3.1.1 File Menu

Use the File menu to create new projects and workspaces, open existing projects and workspaces, or reload existing projects. You can also create objects such as events, procedures, and common data types from the File menu. Commands include:

- New Project...: Creates a new project.
- Open Project...: Opens an existing project. In the Open dialog, select the directory and project, then click **Open**.
- New Workspace...: Creates a new workspace.
- Open Workspace...: Opens an existing workspace. In the Open dialog, select the directory and workspace, then click **Open**.

- **Reload Project:** Reloads a project. When Reload Project is selected, a list of current projects is displayed. Select the project to Reload from the list.
- **Migrate:** Migrates objects from one repository to another. This functionality can be used to migrate metadata from a development to a production integration system.
- **New:** Creates a new object in iStudio. When New is selected, a list of available objects is displayed. Select the object to create. If some objects are grayed-out, then they are not allowed to be created.
- **Sync Adapters:** Pushes metadata to adapters. Adapters can be configured to cache metadata locally so that they avoid any runtime performance penalties around repository access. If they are so configured, any changes made in iStudio will need to be explicitly pushed to the corresponding adapters to refresh their local cached repository metadata. This functionality provides that explicit mechanism to refresh adapter-cached metadata.

Note: Before you attempt to sync adapters, you must ensure that there are no messages flowing between the relevant adapters.

- **Exit:** Exits iStudio.

2.3.1.2 Edit Menu

Use the Edit menu to edit, copy, or delete selected objects. If an object is selected and the Edit menu is not available, that selected object cannot be edited. Commands include:

- **Edit:** Edits a selected object. The type of editing depends on the object selected.
- **Copy:** Copies a selected object.
- **Delete:** Deletes a selected object.
- **Rename:** Renames a selected application.
- **Version:** Creates a new version of or load a selected object.
- **Domain Value Map:** Adds or removes applications from a domain value map.
- **Cross Reference Table:** Adds or removes applications from a cross reference table.
- **Workflow:** Deploys events to Oracle Workflow or edits Oracle Workflow configuration information.

2.3.1.3 Procedure Menu

Use the Procedure menu to invoke or implement procedures. Commands include:

- **Invoke:** Invokes a selected procedure by launching the Invoke Wizard.
- **Implement:** Implements a selected procedure by launching the Implement Wizard.

2.3.1.4 Event Menu

Use the Event menu to publish or subscribe events. Commands include:

- **Publish:** Publishes a selected event by launching the Publish Wizard. An event must be created.
- **Subscribe:** Subscribes to a selected event by launching the Subscribe Wizard. An event must be created.










2.3.1.5 Help Menu



The Help menu provides links to online help. Commands include:

- Contents: Opens the User's Guide.
- About...: Displays version information for iStudio.

2.3.2 Toolbar

The toolbar consists of icons that represent frequently used commands. To display a caption describing the icon, place the cursor on the icon. The following functions are provided:

Function	Icon	Description
New Project		Creates a new project in iStudio.
Open Project		Opens an existing project in iStudio.
Create Integration Object		Creates a new integration object.
Create Like		Creates a new integration object similar to an existing object. This icon is enabled only when an object is selected in the Navigator.
Edit Integration Object		Edits a selected integration object.
Delete Integration Object		Deletes a selected integration object. This icon is enabled only when an integration object is selected in the Navigator.
Publish Event		Publishes a selected event. This icon is enabled only when an integration object is selected in the Navigator.
Subscribe Event		Subscribes to a selected event. This icon is enabled only when an integration object is selected in the Navigator.
Invoke Procedure		Invokes a selected procedure. This icon is enabled only when an integration object is selected in the Navigator.

Function	Icon	Description
Implement Procedure		Implements a selected procedure. This icon is enabled only when an integration object is selected in the Navigator.
Help		Displays the help file.

2.3.3 Design Navigation List

The Design Navigation list displays the hierarchical layout of all objects used in the design phase of an opened project. Each object type in the Deploy Navigation list is identified by an icon and name. A container is represented by a folder icon and is a logical grouping of a specific type of object, such as a Business Object and an Application Data Type.

The objects are grouped as follows:

- Common Views
- Applications
- Workflow
- Enabling Infrastructure

2.3.4 Deploy Navigation List

The Deploy Navigation list displays the hierarchical structure of all objects used in the deploy phase of an opened project. Each object type in the Deploy Navigation list is identified by an icon and name. A container is represented by a folder icon and is a logical grouping of a specific type of object, such as Process Bundles.

The objects are grouped as follows:

- Applications
- Workflow

2.3.5 Context Menus

You can right-click an object to display a context menu, that is, a shortcut menu related to the object.

Navigation List	Selected Item	Context Menu Options
Design	Object, such as Common View, Application, Business Objects, and Common Data Types	New, Edit, Copy, Delete
	Container object, such as an existing event or procedure	New, Edit, Copy Delete, Load Version, New Version
	Workflow object	New, Edit, Copy, Delete, Launch WF Builder, Launch WF Home Page
Deploy	Object such an Application	New, Edit, Copy, Delete, Deploy, Export PL/SQL

Navigation List	Selected Item	Context Menu Options
	Workflow object	New, Edit, Copy, Delete, Deploy, Edit Configuration, Launch WF Home Page, Export
	Container object, such as an existing routing object	New, Edit, Copy, Delete, Create Partition

2.3.6 Detail View

To the right of the Navigation list is the Detail View, composed of one or more property sheets displaying information about the object selected. Often, these property sheets may be edited.

2.4 Using Workspaces in iStudio

A workspace stores user settings and preferences, such as application login credentials and information about the last opened project. Inside a workspace, users can work on multiple projects.

2.4.1 Creating a New Workspace

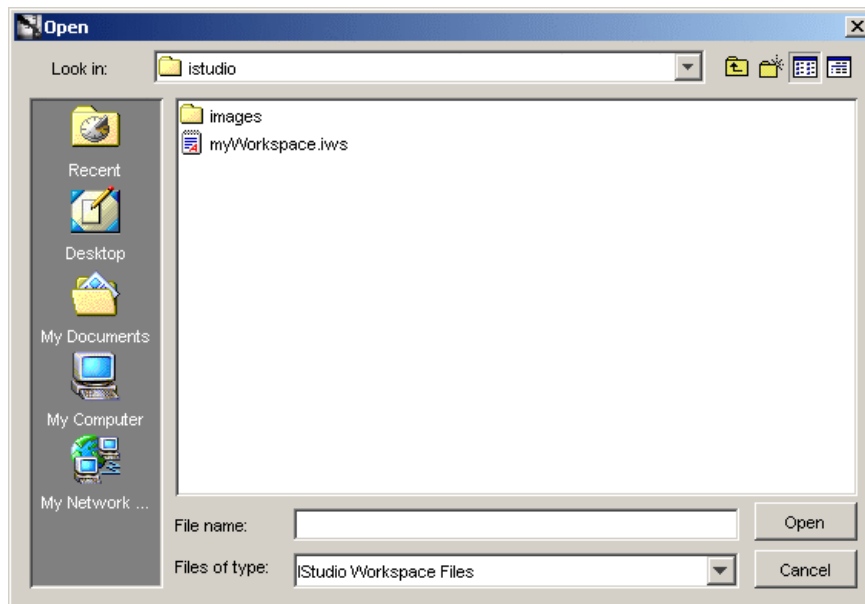
To create a new workspace:

1. From the File menu, select **New Workspace**. The New Workspace Dialog is displayed.
2. Enter a name for the workspace in the Workspace Name field.
3. Click **OK**.

2.4.2 Opening an Existing Workspace

To open an existing workspace:

1. From the File menu, select **Open Workspace**. The Open dialog is displayed.
2. Enter the name and path to an existing workspace or select the workspace to open.



3. Click **Open**. The selected workspace is displayed in iStudio.

2.5 Using Projects in iStudio

A project in iStudio captures all the integration logic for one integration scenario. An integration scenario is defined as a set of two or more applications integrated with each other using OracleAS Integration InterConnect. One project corresponds to one repository. For example, a user may have a development integration environment and a production integration environment. These are two separate projects and must be self-contained in their own separate repositories.

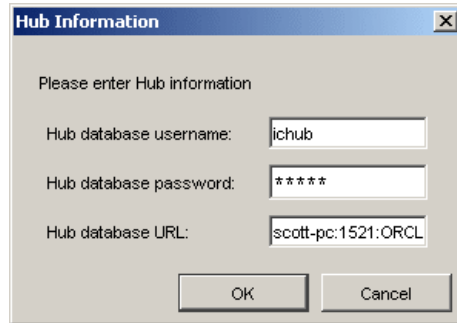
As iStudio is a multiuser tool, multiple users can work on the same project, simultaneously, without jeopardizing the integrity of the metadata.

Note: To create a project in iStudio, the repository must be running.

2.5.1 Creating a New Project

The repository must be running in order to create a project in iStudio. To create a new project in iStudio:

1. From the File menu, select **New Project**. The New Project dialog is displayed.
2. Enter the project name and click **OK**. The Hub Information dialog is displayed.

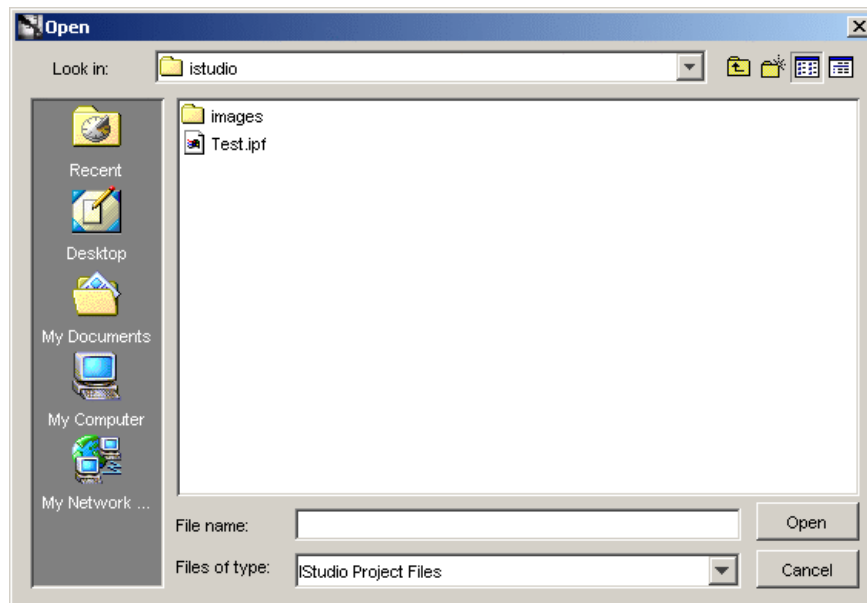


3. Enter information in the following fields:
 - Hub database username: The name of the hub database user. The default username is ichub.
 - Hub database password: The password associated with the hub database user. The default password is set when OracleAS Integration InterConnect is installed.
 - Hub database url: Information of the following format:
machine name:port number:database sid
4. Click **OK**.

2.5.2 Opening an Existing Project

To open an existing project:

1. From the File menu, select **Open Project**. The Open dialog is displayed.



2. Enter the name and path to an existing project or select the workspace to open.
3. Click **Open**. The selected project opens in iStudio.

Creating Applications, Common Views, and Business Objects

This chapter describes how to create and manage applications, common views, and business objects using iStudio. It contains the following topics:

- [Overview of Applications](#)
- [Overview of Common Views and Business Objects](#)

3.1 Overview of Applications

Each component integrated with OracleAS Integration InterConnect is referred to as an application. Each application expresses interest in specific messages, what its internal data type is, and how the message should be mapped to or from that internal type to the external world.

3.1.1 Application View

Each application has its internal data types, formats, and structures that it exposes to the external world. This is the application's public interface. This application interface is called the application view of data. Transformations are used to bridge the gap between application views and common views of data.

Once an application is created in iStudio, it can start participating in the integration scenario by plugging into events and procedures available in the common view.

See Also: [Chapter 4, "Using Events in iStudio"](#) and [Chapter 5, "Using Procedures in iStudio"](#).

3.1.2 Application Data Types

Application Data Types are useful for reusing structure definitions for application views. You can define an Application Data Type once and then utilize it in multiple application views. They are especially useful for defining complex hierarchical data. Application data types have the same function as Common Data Types but relate to a particular application.

See Also: [Section 3.2.1.2, "Creating Common Data Types"](#)

3.1.3 Creating an Application

To create an application:

1. From the File menu, select **New**, then select **Application**. The Create Application dialog is displayed.
2. Enter a name for the application in the Application Name field.
3. Click **OK**.

The application created is displayed in the Design Navigation list under the Applications node.

3.2 Overview of Common Views and Business Objects

The common view is the *hub view* of the integration where each spoke is an application wanting to participate in the integration. After defining a common view by creating a business object and common data types, existing events can be published or subscribed to, and procedures can be invoked or implemented.

See Also: [Chapter 2, "Using iStudio"](#)

3.2.1 Defining Common Views

When defining a common view, you must create business objects and common data types.

3.2.1.1 Creating Business Objects

To create a new business object:

1. From the File menu select **New**, then select **Business Object**. The Create Business Object dialog is displayed.
2. Enter a name for the business object in the Business Object Name field.
3. Click **OK**. The business object is displayed in the Design Navigation list under the Common View node.

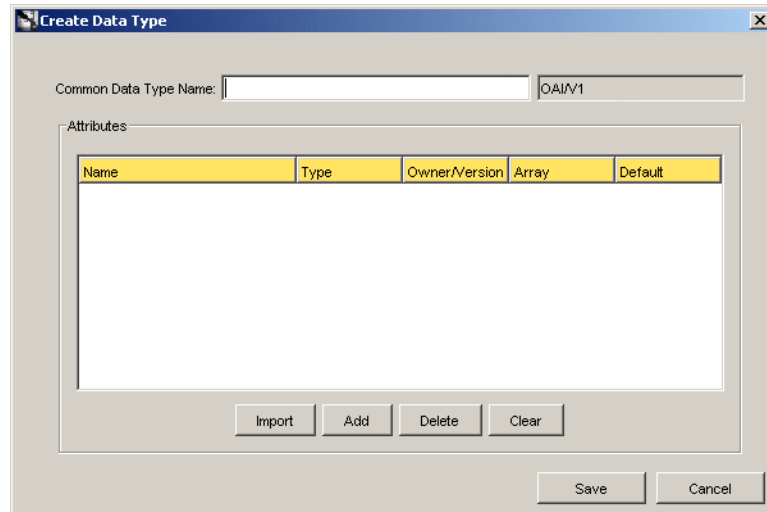
3.2.1.2 Creating Common Data Types

When creating the data associated with an event or a procedure, it is possible to define the data once and reuse it for different integration points. Common data types are used to define such data for reuse and are especially useful for defining complex hierarchical data.

For example, a purchase order contains a header object and an array of line item objects. In addition, the header object contains two address objects: `Bill_To` and `Ship_To`. As a result, the purchase order can be defined once and used for other purchase order-related integration points, such as `Create_Purchase_Order`, `Update_Purchase_Order`, and `Get_Purchase_Order`. Moreover, Address can be defined once and used in the `Bill_To` and `Ship_To` addresses.

To create a common data type:

1. From the File menu, select **New**, then select **Common Data Type**. The Create Data Type dialog is displayed.



Enter a name for the common data type in the Common Data Type Name field.

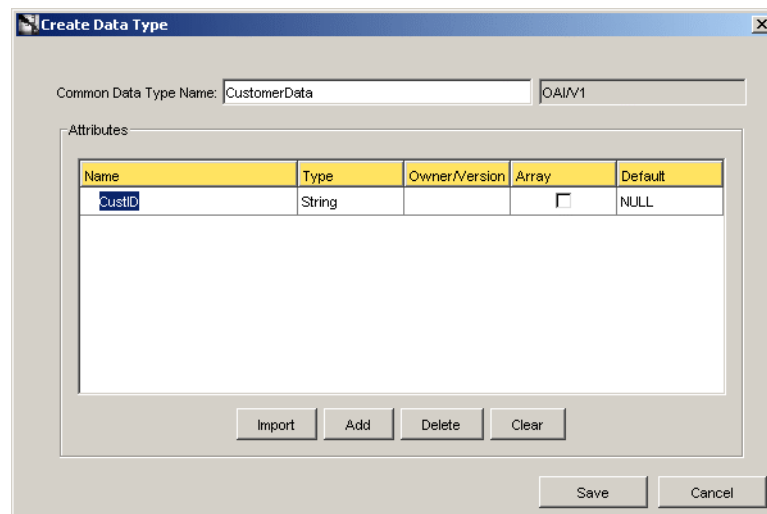
The owner and version number of the common data type display next to the common data type name. This field cannot be edited.

2. Specify the attributes for this common data type using one of the following methods:
 - Add attributes individually.
 - Import attributes from already existing application native data types or APIs.

3.2.1.3 Adding Attributes

To add attributes:

1. In the Create Data Type dialog, click **Add**. A new entry is displayed in the attribute list.



2. Specify the following information by editing the fields.

Name: The name of the attribute.

Type: The type of the attribute. Select the type by selecting the Type column in the attribute entry. A list is displayed. The attribute can be of primitive type such as string, integer, float, double, date, or another common data type used to build hierarchical data types.

Array: Select this check box if the attribute is a collection instead of a single attribute.

Default: The default value of the field if it is not populated at runtime.

3. Click **Save**. Repeat the preceding steps to add other attributes.

3.2.1.4 Importing Attributes

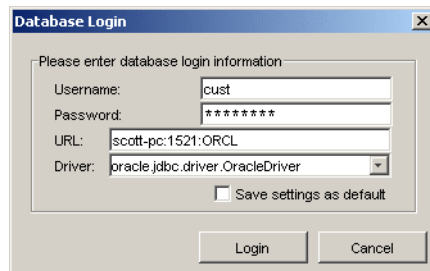
To import attributes:

1. In the Create Data Type dialog, click **Import**. Attributes can be imported from various sources.

The following steps describe the database import facility.

See Also: [Appendix B, "Using the Data Definition Description Language"](#)

2. Click **Database**. The Database Login dialog is displayed.



3. Enter information in the following fields:

User Name: The database log in name.

Password: The database log in password.

URL: The computer name: port number: database SID.

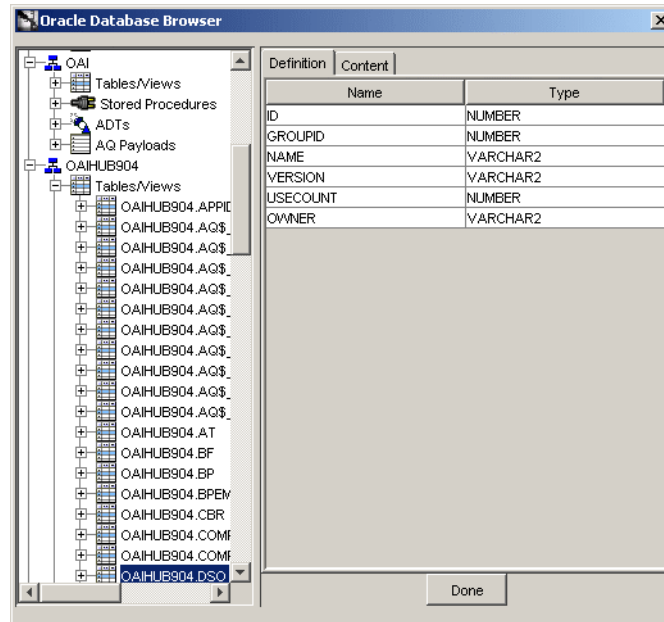
Driver: The JDBC driver used to connect to the database.

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

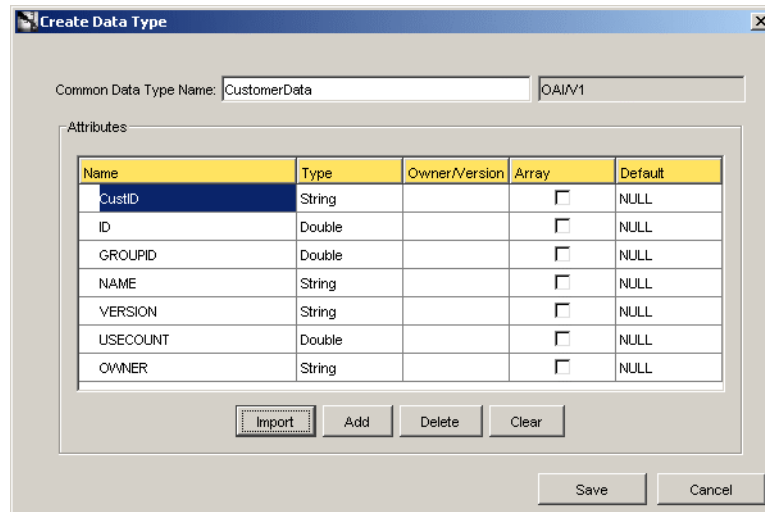
4. Click **Login**.

After logging in, the database tables and arguments display in the Database Browser window.

Select the fields to add. To select a range of fields, press Shift when clicking the mouse button. To select multiple items, press Ctrl while clicking the mouse button.



5. Click **Done** to import the attributes into the common data type. The selected attributes are displayed in the Create Data Type dialog.



3.2.1.5 Deleting and Clearing Attributes

To delete a selected attribute:

- In the Create Data Type dialog, select the attribute to be deleted, and click **Delete**.

To clear all attributes:

- In the Create Data Type dialog, click **Clear**.

Using Events in iStudio

This chapter describes how to use iStudio to create, publish, and subscribe to events. It contains the following topics:

- [Overview of Events](#)
- [Creating Events](#)
- [Publishing and Subscribing to an Event](#)

4.1 Overview of Events

An event is an integration point used to model the Publish/Subscribe paradigm. An event has associated data that is the common view of all the data to be exchanged through this event.

The data structure that should be used for defining the common view is entirely dependent on the integration scenario, and the choice is left to the implementor. The only condition that data structure should satisfy is that it should be a superset of all the application views for applications that will be publishing or subscribing to this event. The choices for common view data include:

- Corporate internal standards enforced by the organization where OracleAS Integration InterConnect is being implemented.
- Industry standard definitions, such as Open Applications Group (OAG) and Business Object Definitions (BOD).
- One of the application views can be used as the common view, if the scenario has one application that is driving the integration.

For example, App1 and App2 publish customer names, and App3 subscribes to it. If App1 publishes `First Name`, `Last Name`, and `Middle Initial`, and App2 publishes `First Name`, `Last Name`, `Prefix`, and `Suffix`, the event could be defined as follows:

```
New Customer Event
Prefix
First Name
Last Name
Middle Initial
Suffix
```

See Also: ["How the Hub-and-Spoke Methodology Works"](#) on page 1-6

4.1.1 Event Maps

If an application publishes exactly the same data structure for two or more events, event maps allow OracleAS Integration InterConnect to distinguish which message corresponds to which event. For example, an application publishes the same Customer Application Data Type whether or not it is a `Create_Customer` event or an `Update_Customer` event. Through event maps, OracleAS Integration InterConnect can determine which messages correspond to `Create_Customer` and `Update_Customer`.

Note: Event maps need to be used only if two or more events published by a particular application have the exact same application view structure.

4.2 Creating Events

To create an event:

1. From the File menu, click **New**, then select **Event**. The Create Event dialog is displayed.

Name	Type	Owner/Version	Array	Default
------	------	---------------	-------	---------

2. Enter the following information in the fields:
 - **Business Object:** The name of the category to which the event belongs. Select a category from the list.
 - **Event Name:** The name of the event. Only alphanumeric characters can be used.
 - **OAI/V1:** The owner and version number of the Business Object. This field cannot be edited.
3. Add or import attributes to this event.
4. Click **Save**.

See Also:

- "Adding Attributes" on page 3-3
- [Appendix B, "Using the Data Definition Description Language"](#)
- "Deleting and Clearing Attributes" on page 3-5

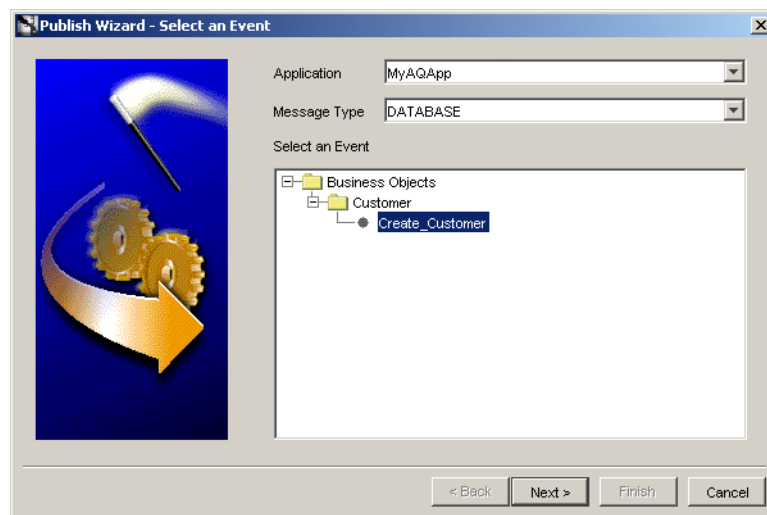
4.3 Publishing and Subscribing to an Event

The publish/subscribe paradigm is used for asynchronous one-way communication. The sending application is said to publish the event. The receiving application subscribes to the event.

4.3.1 Publishing an Event

Publishing an event in an application in iStudio involves the Publish Wizard. To start the Publish Wizard:

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

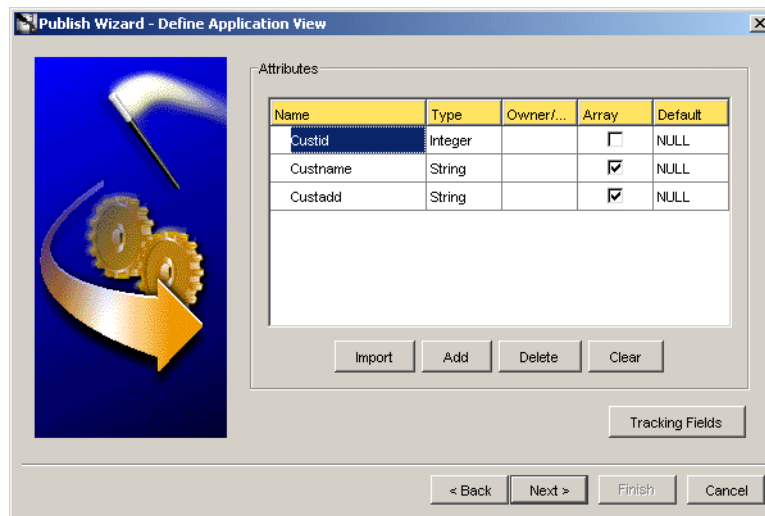


- a. Enter following information in the fields:
 - Application: The name of the invoking application is selected by default.
 - Message Type: The mode of communication between OracleAS Integration InterConnect and the application. Select from the following message types:
 - Database: OracleAS Integration InterConnect communicates with the application using the database.
 - Generic: OracleAS Integration InterConnect communicates with the application using a user-defined bridge.
 - XML: OracleAS Integration InterConnect communicates with the application using XML data described through a data type definition (DTD) using the FTP, SMTP, HTTP, MQ Series, or user-defined adapters.

- AQ: OracleAS Integration InterConnect communicates with the application through Oracle Advanced Queues using the Advanced Queue adapter. The payload can be Oracle Objects where fields may be XML or RAW XML.
- D3L: OracleAS Integration InterConnect communicates with the application using non-XML data formats described through D3L using the FTP, SMTP, HTTP, and MQSeries adapters.

Note: Other choices will be visible if you've purchased and installed additional adapters for Oracle E-Business Suite, SAP, Peoplesoft, and Siebel.

- b. Select the event name.
 - c. Click **Next**.
2. The Define Application View page is displayed.

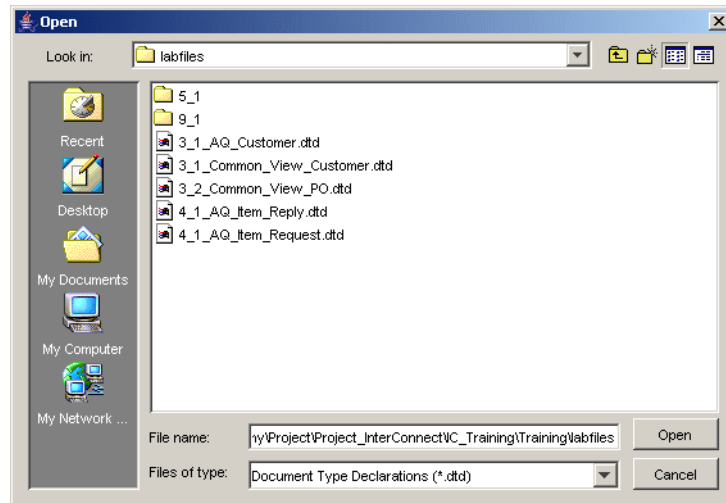


Once an event is selected to publish, the application view is defined. The application view page is initially an empty table. Define the attributes using Add, or import the definitions from a database or an API Repository using Import.

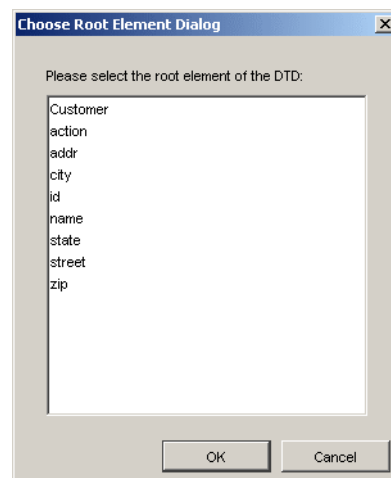
- a. Click **Add** or **Import** to add or import attributes.

See Also:

- ["Adding Attributes"](#) on page 3-3
 - ["Importing Attributes"](#) on page 3-4
 - ["Deleting and Clearing Attributes"](#) on page 3-5
 - [Appendix B, "Using the Data Definition Description Language"](#)
- b. To import an XML DTD, click **Import** and select XML to display a file dialog:

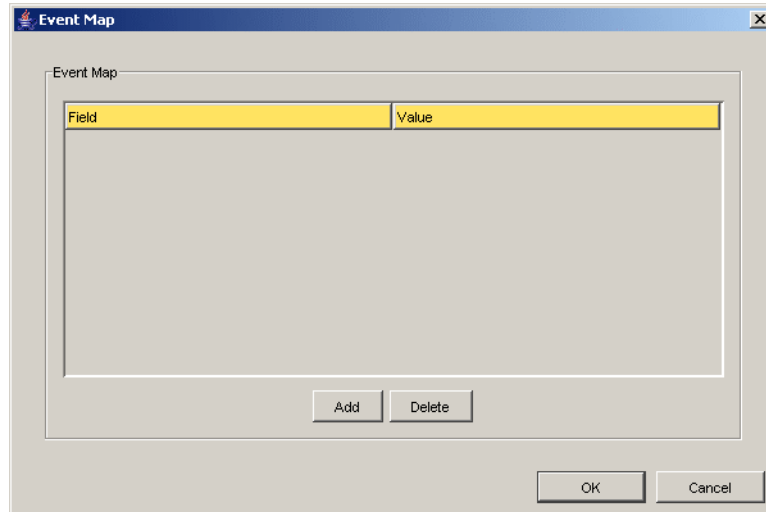


- c. Select a DTD file and click **Open**. The Choose Root Element dialog is displayed.

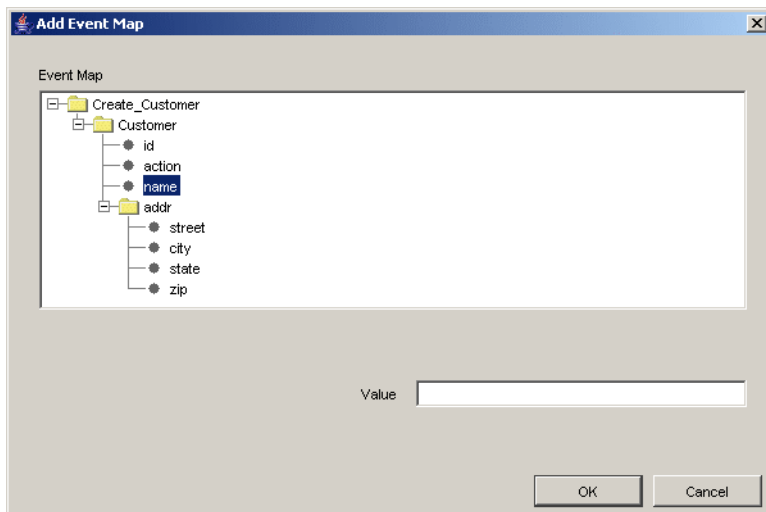


- d. Select a root DTD element and click **OK**.
- e. If this is a XML type message, the Event Map button is enabled. To define the event map, click **Event Map**. The Event Map dialog is displayed.

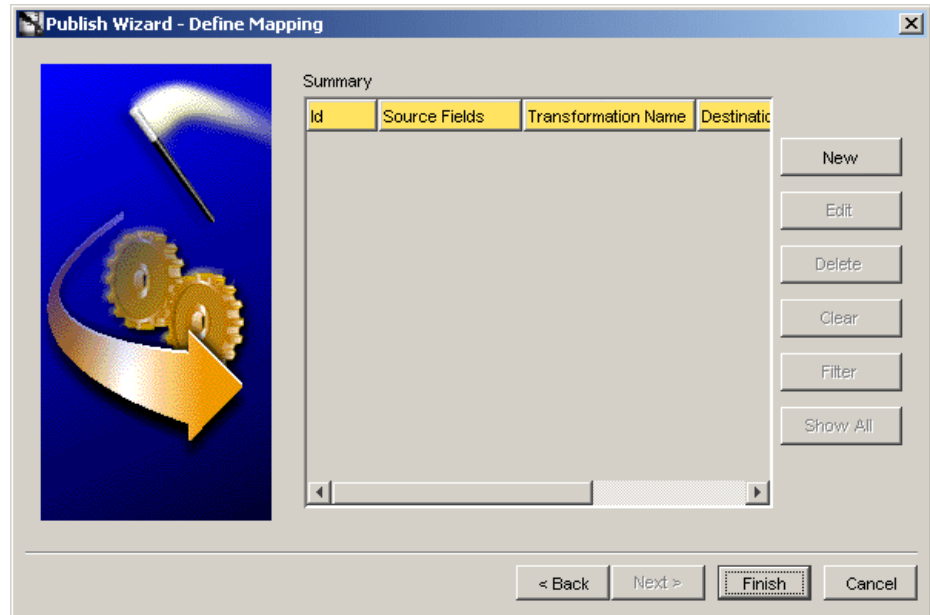
Note: Event maps need to be used *only* if two or more events published by a particular application have the exact same application view structure.



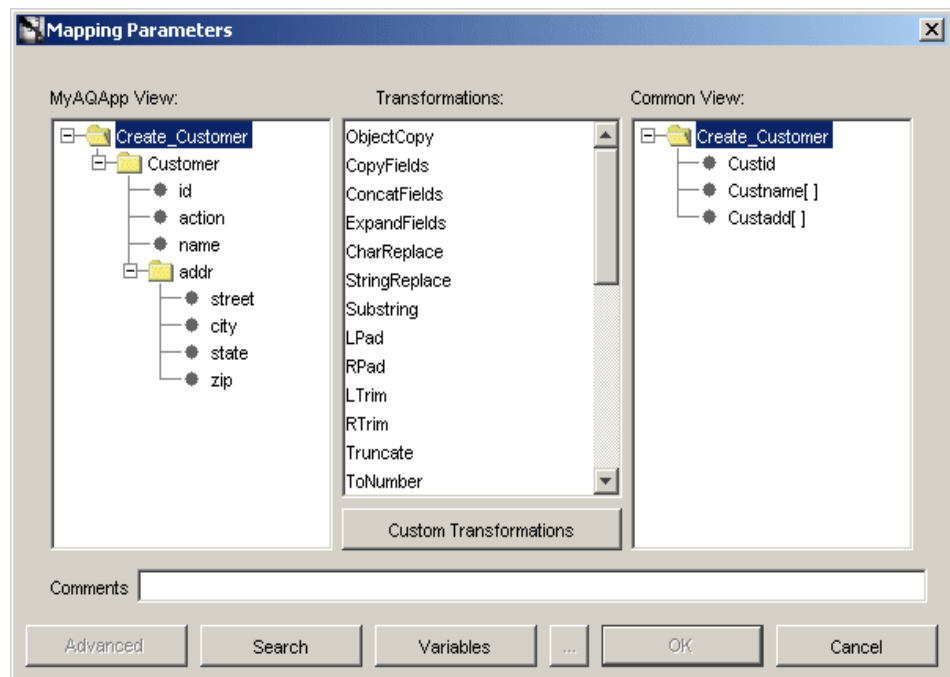
- f. Click **Add** to add an event map attribute. The New Event Map dialog is displayed.



- g. Expand the list and select an attribute and enter a value in the Value field.
 - h. Click **OK** on the Add Event Map dialog to return to the Event Map dialog.
 - i. Click **OK** to return to the Publish Event wizard.
 - j. Click **Next**.
3. Click **Next** on the Define Application View page. The Define Mapping page is displayed. Mapping involves copying the individual fields or simple shape-change transformations.



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



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

See Also: ["Invoking a Procedure"](#) on page 5-3

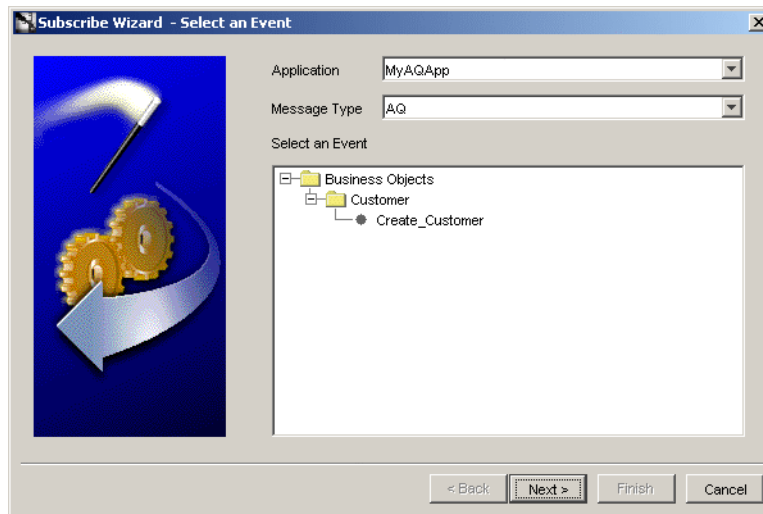
- b. Click **OK** to return to the Publish Event Wizard.

4. Click **Finish**.

4.3.2 Subscribing to an Event

Subscribing to an application event in iStudio involves using the Subscribe Wizard. To subscribe to an event in an application:

1. In the Design Navigation list, expand the Application node. Select and expand the Application node to display the Subscribed Events leaf. Right-click **Subscribed Events** and select **Subscribe**. The Subscribe Wizard is displayed.



- a. Enter following information in the fields:

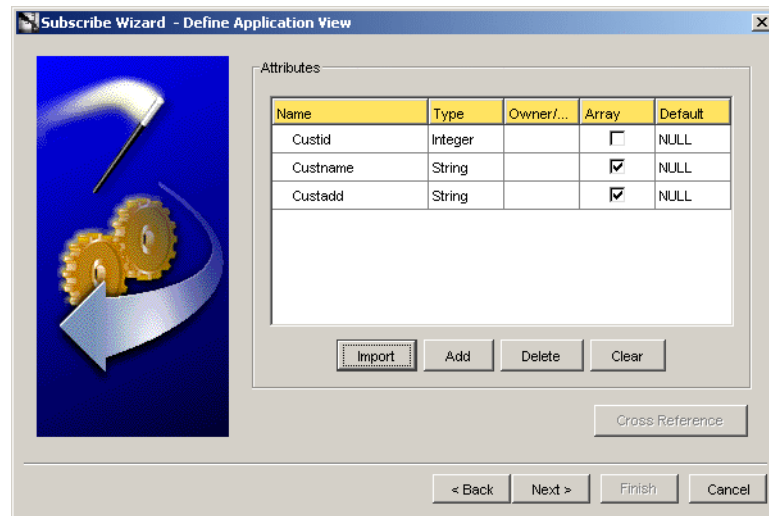
Application: The name of the application selected in the navigation list, which invokes the procedure, appears selected by default.

Message Type: The message type that specifies the mode of communication between OracleAS Integration InterConnect and the application. Select from the following message types:

- **Database:** OracleAS Integration InterConnect communicates with the application using the database.
- **Generic:** OracleAS Integration InterConnect communicates with the application using a user-defined bridge.
- **XML:** OracleAS Integration InterConnect communicates with the application using XML data described through a data type definition (DTD) using the FTP, SMTP, HTTP, MQ Series, or user-defined adapters.
- **AQ:** OracleAS Integration InterConnect communicates with the application through Oracle Advanced Queues using the Advanced Queue adapter. The payload can be Oracle Objects where fields may be XML or RAW XML.
- **D3L:** OracleAS Integration InterConnect communicates with the application using non-XML data formats described through D3L using the FTP, SMTP, HTTP, and MQSeries adapters.
- **Oracle Applications:** OracleAS Integration InterConnect communicates with Oracle Applications using this message type.

Note: Other choices will be visible if you've purchased and installed additional adapters for SAP, Peoplesoft, and Siebel.

- b. Select the event name.
 - c. Click **Next**.
2. The Define Application View page is displayed.

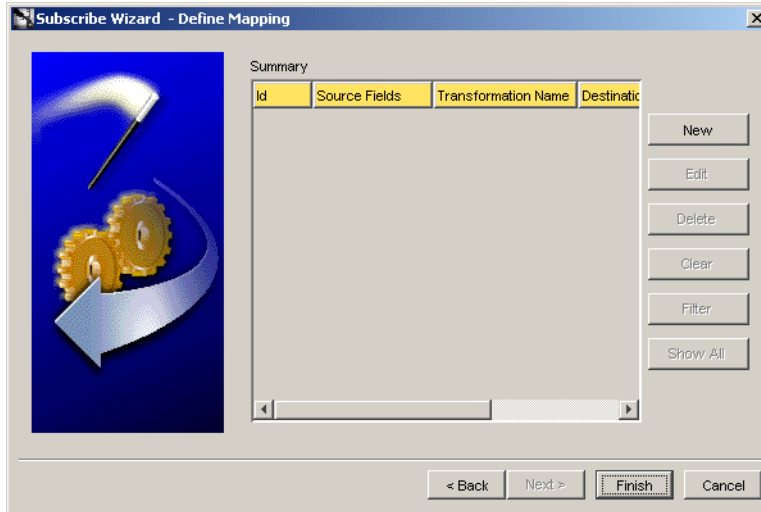


Once an event is selected to subscribe to, the application view is defined. The application view page is initially an empty table. Define the attributes using **Add** or import the definitions from a database or an API Repository using **Import**.

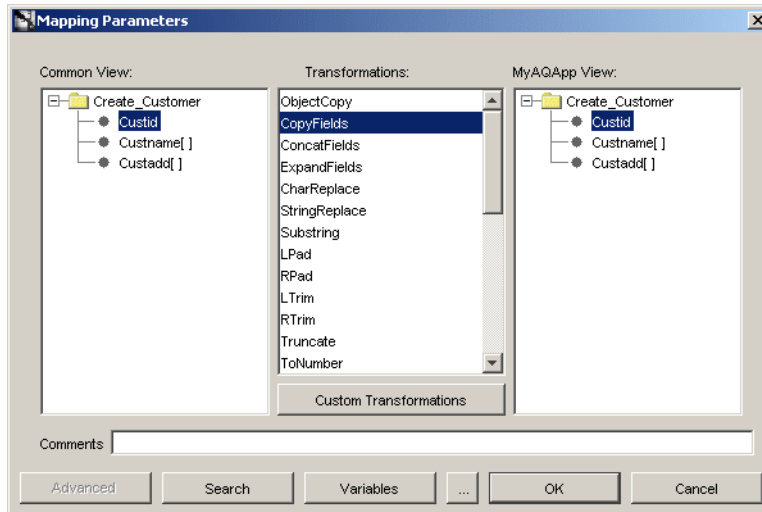
- a. Click **Add** or **Import** to add or import attributes.

See Also:

- ["Adding Attributes"](#) on page 3-3
 - ["Importing Attributes"](#) on page 3-4
 - ["Deleting and Clearing Attributes"](#) on page 3-5
 - [Appendix B, "Using the Data Definition Description Language"](#)
- b. Populate and look up cross-reference tables by clicking **Cross Reference...** The Cross Reference dialog is displayed. Click **OK** to return to the Subscribe Wizard.
3. Click **Next** on the Define Application View page. The Define Mapping Page is displayed. Mapping can involve either copying the individual fields or simple shape-change transformations.



- a. Click **New** to define mappings. The Mapping Parameters dialog is displayed.



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

See Also: ["Invoking a Procedure"](#) on page 5-3

- b. Click **OK** to return to the Subscribe Event Wizard.
- 4. Click **Finish**.

Using Procedures in iStudio

This chapter describes using iStudio to create, invoke, and implement procedures. It contains the following topics:

- [Using Procedures](#)
- [Invoking and Implementing a Procedure](#)

5.1 Using Procedures

A procedure is an integration point used to model the Request/Reply paradigm. The request/reply paradigm is used for two-way context sensitive communication. This communication can be either synchronous (the requesting application is blocking until it receives a reply) or asynchronous (the requesting application gets the reply asynchronously, it does not block-wait for the response after sending the request). This is a modeling paradigm only, no actual procedures are called. An application can either invoke a procedure to model sending a request and receiving a reply, or implement a procedure to model receiving a request and sending a reply. Similar to events, a procedure has associated data. While an event is only associated with one data set, a procedure has two data sets: one for the request, *IN* data and one for the reply, *OUT* data.

Note: Synchronous request/reply can only be used if an application supports an outbound synchronous interface. Currently, only the Oracle database adapter qualifies for such support. For all other adapters, only asynchronous request/reply is available. This is a limitation of the protocols exposed by systems to communicate.

For example, if a `Get_Address` procedure is defined so that the request contains the social security number, *SSN*, for a person and the reply contains the address in four fields: *Street*, *City*, *Zip*, *State*, then the procedure is defined as follows:

```
get Address Procedure
SSN IN
Street OUT
City OUT
Zip OUT
State OUT
```

The data structure that should be used for defining the common view is entirely dependent on the integration scenario, and the choice is left to the implementor. The only condition that data structure should satisfy is that it should be a superset of all

the application views for applications that will be publishing or subscribing to this event. The choices for common view data include:

- Corporate internal standards enforced by the organization where OracleAS Integration InterConnect is being implemented.
- Industry standard definitions, such as Open Applications Group (OAG) and Business Object Definitions (BOD).
- One of the application views can be used as the common view, if the scenario has one application that is driving the integration.

See Also: ["How the Hub-and-Spoke Methodology Works"](#) on page 1-6

5.1.1 Creating a Procedure

To create a procedure:

1. From the File menu, select **New**, and then select **Procedure**. The Create Procedure dialog is displayed.

Name	Type	Owner/Version	Array	Default	IN/OUT/NO...

2. Enter required information in the following fields:
 - Business Object Name: The name of the category to which the procedure belongs. Select BO name from the list.
 - Procedure Name: The name of the procedure. Only alpha-numeric characters can be used.
 - OAI/V1: The owner and version number of the procedure. This field cannot be edited.
3. Add or import attributes to the procedure.
4. Click **Save**.

See Also:

- "Adding Attributes" on page 3-3
- "Importing Attributes" on page 3-4
- "Deleting and Clearing Attributes" on page 3-5
- Appendix B, "Using the Data Definition Description Language"

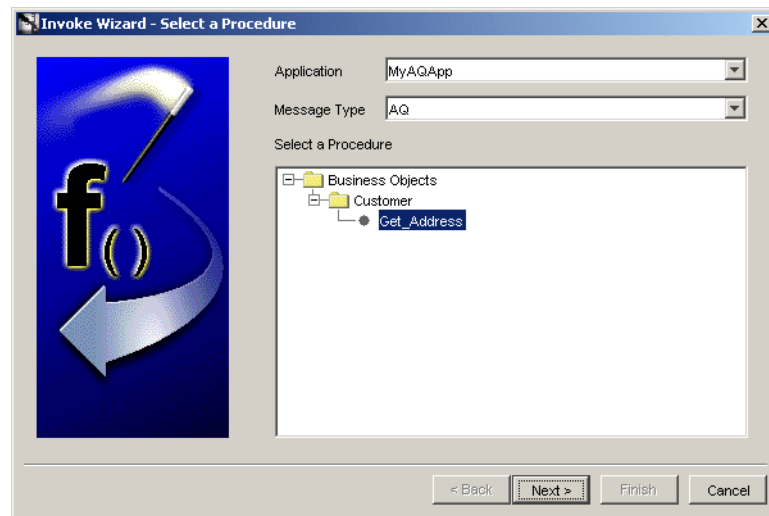
5.2 Invoking and Implementing a Procedure

Procedures are used to model the request/reply messaging paradigm. The requesting application invokes the procedure. The replying application implements the procedure.

5.2.1 Invoking a Procedure

Use Invoke Wizard to invoke a procedure in iStudio. To start the Invoke Wizard:

1. In the Design Navigation list, expand the Application node.
2. Select and expand the Application node to display the Invoked Procedures leaf.
3. Right-click **Invoke Procedures**, and select **New**. The Invoke Wizard is displayed.

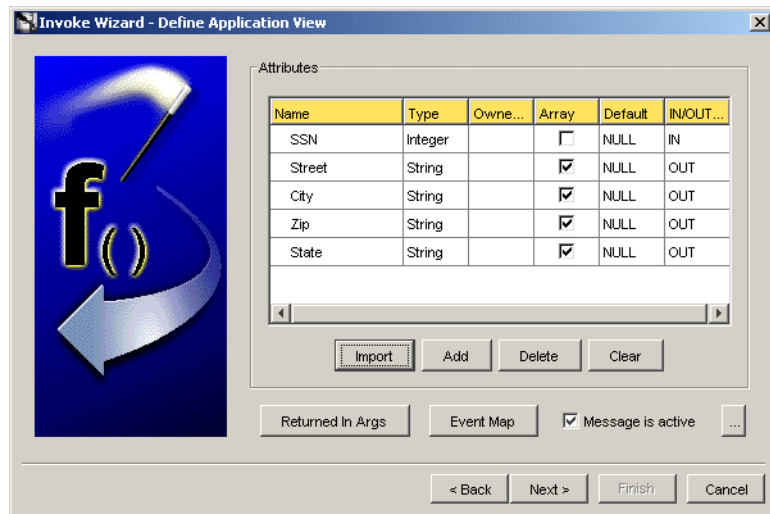


4. Enter the following information in the fields:
 - Application: the name of the invoking application is selected by default.
 - Message Type: The mode of communication between OracleAS Integration InterConnect and the application. Select one of the following message types:
 - Database: OracleAS Integration InterConnect communicates with the application using the database.
 - Generic: OracleAS Integration InterConnect communicates with the application using a user-defined bridge.
 - XML: OracleAS Integration InterConnect communicates with the application using XML data described through a data type definition (DTD) using the FTP, SMTP, HTTP, MQ Series, or user-defined adapters.

- D3L: OracleAS Integration InterConnect communicates with the application using non-XML data formats described through D3L using the FTP, SMTP, HTTP, and MQSeries adapters.

Note: Other choices will be visible if you've purchased and installed additional adapters for Oracle Applications, SAP, Peoplesoft, and Siebel.

- AQ: OracleAS Integration InterConnect communicates with the application through Oracle Advanced Queues using the Advanced Queue adapter. The payload can be RAW XML or Oracle Objects where fields may be XML .
5. Select the procedure to invoke in the Select a Procedure box.
 6. Click **Next** in the Select a Procedure page. The Define Application View page is displayed.



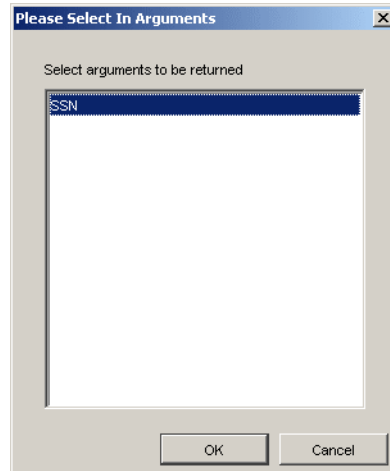
Once a procedure is selected to invoke, the application view is defined. The application view page is initially an empty table.

7. Define the attributes using Add or import the definitions from a database or an API Repository using Import.

See Also:

- ["Adding Attributes"](#) on page 3-3
- ["Importing Attributes"](#) on page 3-4
- ["Deleting and Clearing Attributes"](#) on page 3-5
- [Appendix B, "Using the Data Definition Description Language"](#)

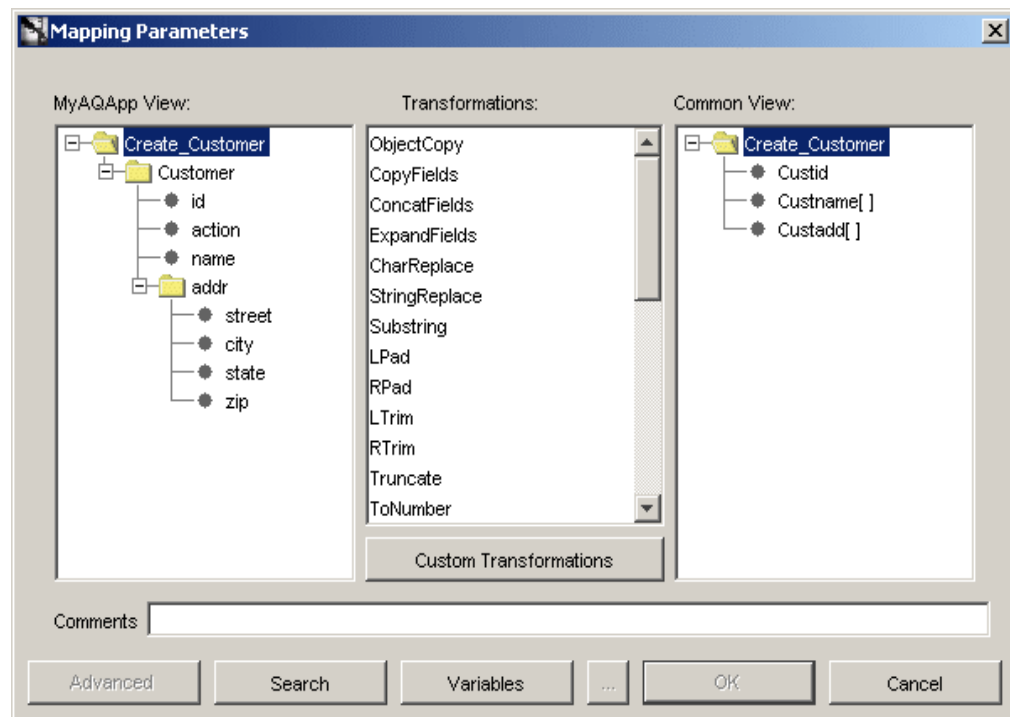
8. Click **Returned In Args** to specify IN arguments to be returned. The Please Select In Arguments dialog is displayed.



9. Select the input and output arguments to be returned. Use the left mouse button to select multiple arguments. Only non user-defined input arguments are shown for selection.
10. Click **OK** to return to the Define Application View page.
11. Click **Next**. The Define Mapping IN Arguments Page is displayed.

Mapping arguments involves copying the individual fields or simple shape-change transformations.

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



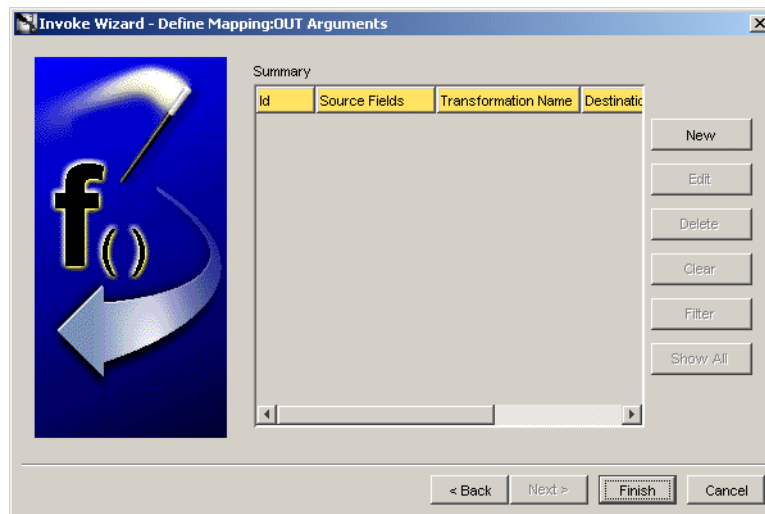
To map fields in the application view to fields in the common view, use a transform. To use a transform to map fields,

1. Select fields to map from in the application view. Use the left mouse button to select multiple fields in a view.
2. Select the transformation, for example, `ConcatFields`.
3. Select the fields to map to in the common view. Use the left mouse button to select multiple fields in a view.
4. Click **Apply** to confirm selection and continue specifying additional mappings.
5. When all mappings have been made, click **OK**.

See Also: ["Invoking a Procedure"](#) on page 5-3

13. Click **Next**. The Define Mapping OUT Arguments Page is displayed.

Mapping arguments involves copying the individual fields or simple shape-change transformations. Use this page to map the common view return arguments to the application view return arguments.



14. Click **New** to define mappings.

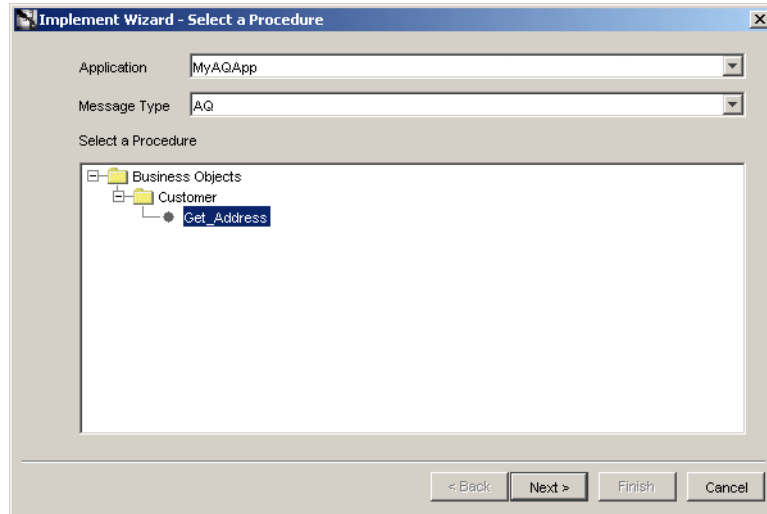
See Also: Step 6 on page 5-4

15. Click **Finish**.

5.2.2 Implementing a Procedure

Implement a procedure in iStudio using the Implement Wizard. To start the Implement Wizard:

1. In the Design navigation list, expand the Application node. Select and expand the Application node to display the Implemented Procedures leaf. Right-click **Implemented Procedures**, and select **New**. The Implement Wizard is displayed.



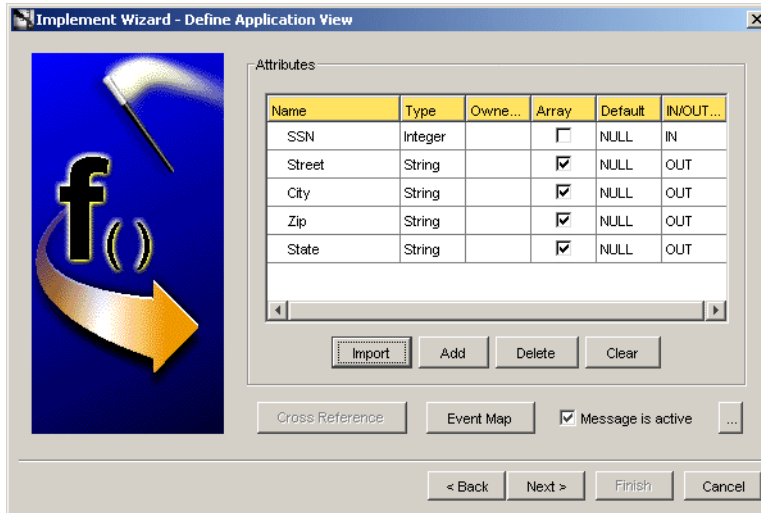
Use this page to select a procedure to implement.

2. Select information for the following fields:

- **Application:** The name of the application selected in the navigation list, which invokes the procedure, appears selected by default. Select an application from the list.
- **Message Type:** This field specifies the mode of communication between OracleAS Integration InterConnect and the application. Select from the following message types:
 - **Database:** OracleAS Integration InterConnect communicates with the application using the database.
 - **Generic:** OracleAS Integration InterConnect communicates with the application using a user-defined bridge.
 - **XML:** OracleAS Integration InterConnect communicates with the application using XML data described through a DTD using the FTP, SMTP, HTTP, MQ Series, or user-defined adapters.
 - **AQ:** OracleAS Integration InterConnect communicates with the application through Oracle Advanced Queues using the Advanced Queue adapter. The payload can be Oracle Objects where fields may be XML or RAW XML.
 - **D3L:** The adapter communicates with the application using D3L.

3. Select the procedure to invoke.

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



Initially, this page is an empty table. Attributes can be defined by using Add. Attribute definitions can be imported from a database or an API Repository by using Import.

See Also:

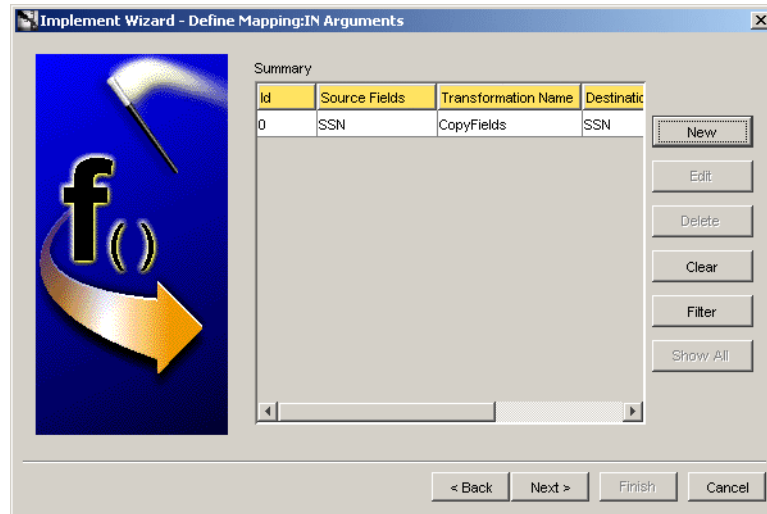
- ["Adding Attributes"](#) on page 3-3
- ["Importing Attributes"](#) on page 3-4
- ["Deleting and Clearing Attributes"](#) on page 3-5
- [Appendix B, "Using the Data Definition Description Language"](#)

5. Click **Cross Reference...** to populate cross reference tables.

See Also: ["Populating Cross-reference Tables"](#) on page 6-7

6. Click **Next**. The Define Mapping IN Arguments page is displayed.

Mapping may involve copying individual fields, or simple shape-change transformations. After clicking next on the Define Application View page, the Define Mapping IN Arguments page is displayed.



7. Click **New** to define IN mappings.

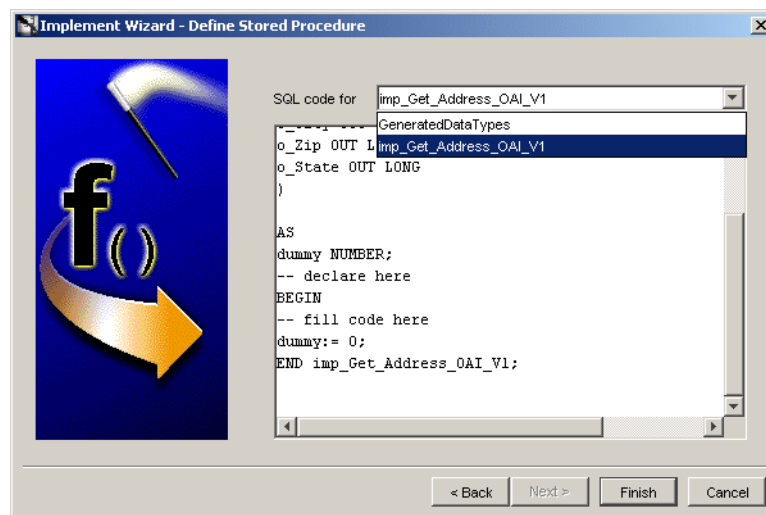
See Also: Step 6 on page 5-4

8. Click **Next**. The Define Mapping OUT Arguments page is displayed.

9. Click **New** to define OUT mappings.

See Also: Step 6 on page 5-4

10. Click **Next**. The Define Stored Procedure page is displayed.



If the message type selected was database, the data is received by a stored procedure. In this stored procedure, the action performed when the values are returned to the application can be specified. The adapter invokes the stored procedure at runtime with the corresponding data.

The following arguments will be returned:

- All OUT arguments.
- All IN arguments specified to be returned as part of the reply.

11. Select a generated procedure from the SQL Code For list.
12. Click **Finish**.

Enabling Infrastructure

This chapter describes the enabling infrastructure tasks in iStudio. It contains the following topics:

- [Enabling Infrastructure](#)
- [Working with Content-based Routing](#)
- [Working with Domain Value Mappings](#)
- [Working with Cross-referencing](#)

6.1 Enabling Infrastructure

Enabling Infrastructure provides additional important features that are critical to enable an end-to-end integration. These features include:

- **Content-based Routing.** Route messages by building business rules based on message content. For example, a procurement system routes fulfillment requests to different fulfillment centers based on an originating location.

Content based routing can be done for both events and procedures and is driven off the common view data. The adapter that publishes an event (or invokes a procedure), evaluates the content based routing rules to determine the recipient list. This recipient list is then added to the JMS header for the message before it is sent off to the OracleAS Integration InterConnect Hub. In the hub, AQ looks up the recipient list and wakes up the relevant target adapters.

Note: When defining content based routing rules, make sure that all routing cases are covered for all possible values of the fields used in the rule. Once you define even one content based routing rule, the default event based routing rules are no longer in effect.

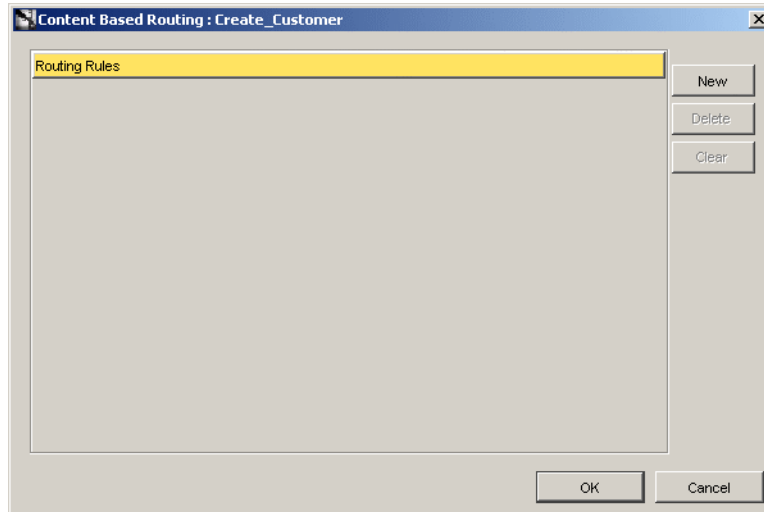
- **Domain Value Mapping.** Map code tables across systems. For example, a purchase order in a procurement system has a PO Status field with domain values, Booked and Shipped. The corresponding field in a fulfillment system has the domain values, 1 and 2. OracleAS Integration InterConnect allows the user to create the mappings booked=1, shipped=2 so that it can correlate these values at runtime without each system understanding the domain value set of the other system.
- **Cross Referencing.** Correlate keys that uniquely identify the entities in one application with corresponding entities created in other applications. For example, a purchase order created in a procurement system has a native ID X. The purchase order is then routed to a fulfillment system, where it is created with native ID Y. As a result, X and Y must be cross referenced for OracleAS Integration

InterConnect to correlate communication about this same logical entity in two different systems without each system understanding the native ID of the other system.

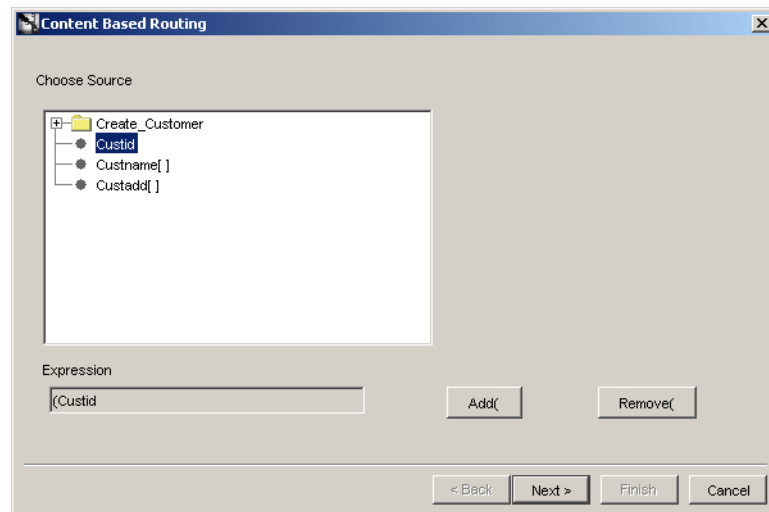
6.2 Working with Content-based Routing

To modify content-based routing for an event or procedure:

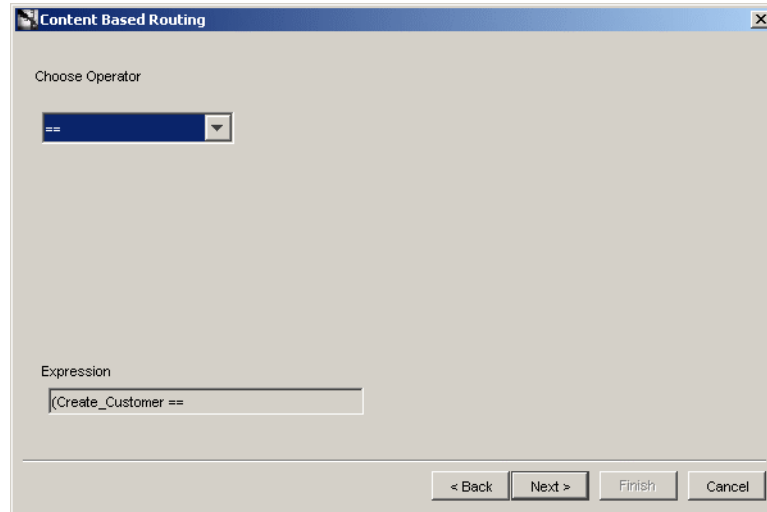
1. Right-click the event or procedure under the Content-based Routing node in the Design Navigation list, and then click **Edit**. The Content Based Routing Rules dialog is displayed.



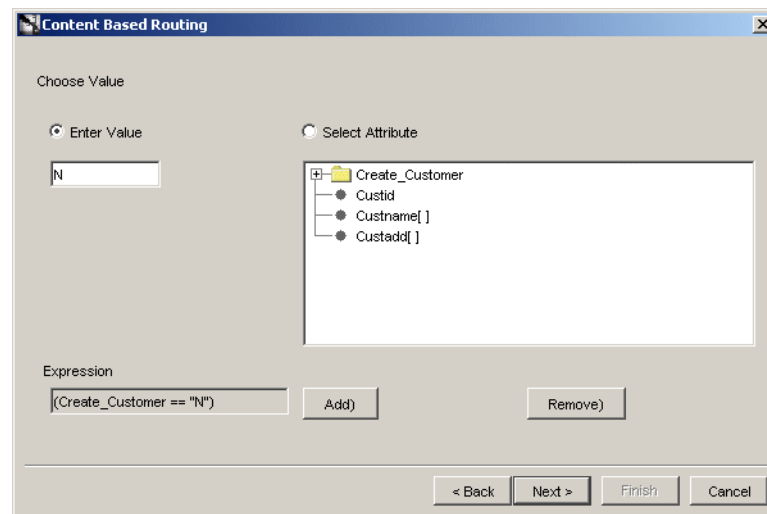
2. Click **New**. The wizard that provides a series of pages to follow for editing content-based routing is displayed.



3. Choose the source event attribute to be used for building rules, and click **Next**. An attribute can be chosen either from the message payload (common view data) or the message header that accompanies this payload.



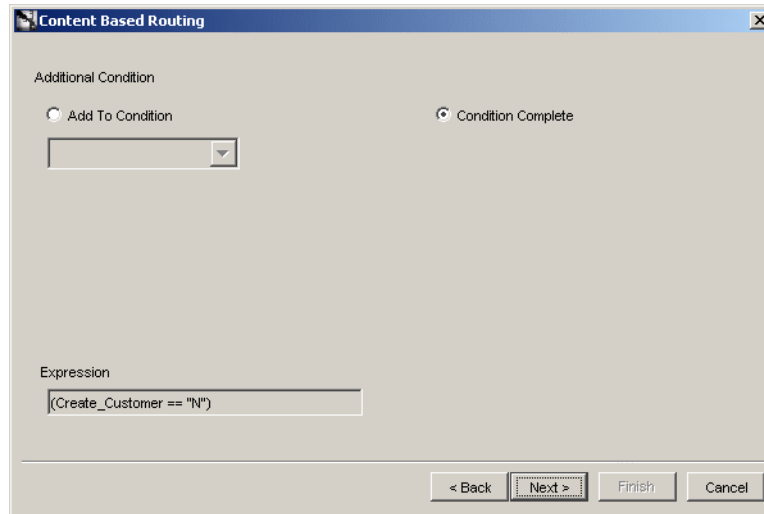
4. Select an operator from the list and click **Next**. The Choose Value page is displayed.



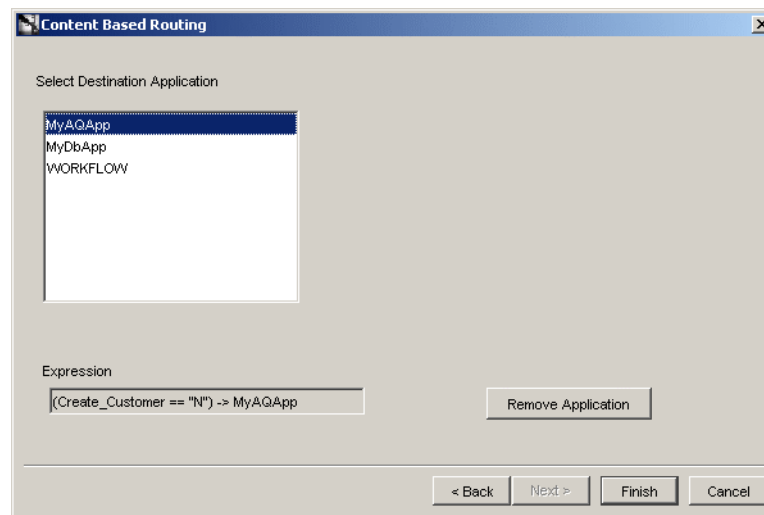
5. Enter a literal value or select another attribute in the message payload (or header) to be compared against the source event attribute selected in Step 3. For literal values, enter a value in the text field. For attributes, select an attribute from the navigation list.

Note: In some cases, in the Choose Value page, a list of available applications is visible. This list appears only when you select the SendingApplication event attribute in the Choose Source page.

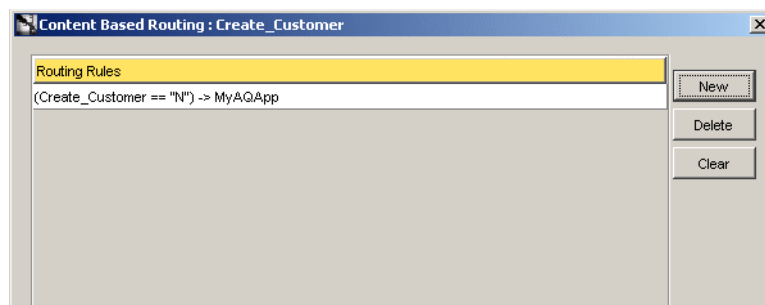
6. Use Select Attribute to compare one value in an attribute to another.
7. Click **Next**. The Additional Condition page is displayed. You can add to the condition through the operators AND and OR.



8. To further build your rule, select **Add To Condition**. This can be used to build complex routing rules such Age < 50 AND Salary > 100000 OR AGE >= 50. Repeat steps 3-7.
9. If your rule is built, select **Condition Complete**, and click **Next**. The Select Destination Application page is displayed.



10. Select one or more applications from the Select Destination Application page, and click **Finish**.



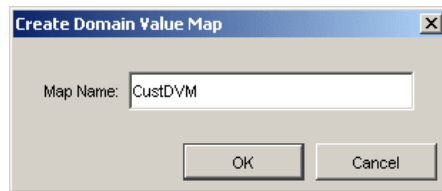
The Content Based Routing Rule is created and displayed in the Content Based Routing dialog.

11. Click **New** to add another rule or click **OK** to finish.

6.3 Working with Domain Value Mappings

To create a domain value mappings table:

1. In iStudio, click **Domain Value Maps** under Enabling Infrastructure.
2. Select **New** from the File menu, and select **Domain Value Mapping**. The Create Domain Value Mapping dialog is displayed.

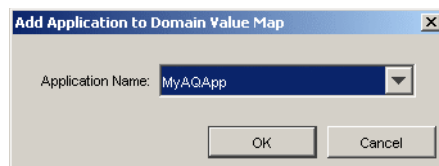


3. Enter a name for the domain value map in the Map Name field.
4. Click **OK**.

6.3.1 Adding Applications to Domain Value Mappings

To add applications to domain value mappings:

1. In the Design Navigation list, select the domain value mapping and right-click.
2. From the context menu, select **Add Application**. The Add Application to Domain Value Map dialog is displayed.

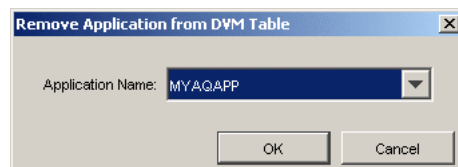


3. Select an application name from the list.
4. Click **OK**.

6.3.2 Removing Applications From Domain Value Mappings

To remove applications from the domain value mappings:

1. In the Design Navigation list, select a domain value mapping and right-click.
2. From the context menu, select **Remove Application**. The Remove Application from Domain Value Mapping dialog is displayed.

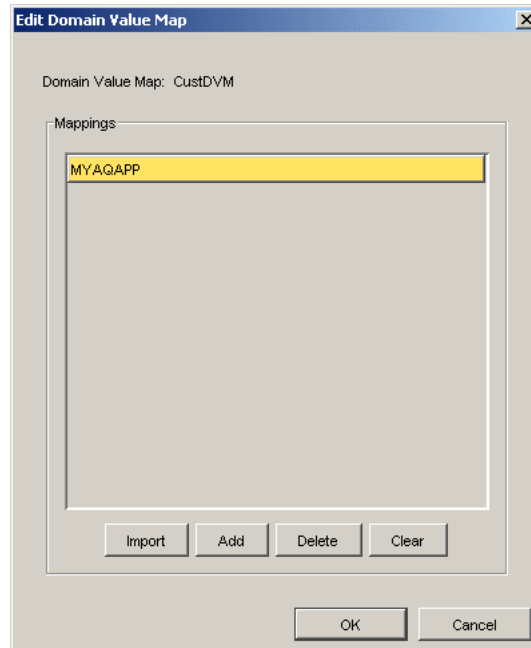


3. Select the Application Name to remove from the list.
4. Click **OK**.

6.3.3 Modifying Domain Value Mappings

To modify data domain value mappings:

1. In the Design Navigation list, select a domain value mapping and right-click.
2. From the context menu, select **Edit Values**. The Edit Domain Value Map dialog appears.



3. Click **Add** to add mappings or **Import** to import mappings.
4. Click **OK**.

6.3.4 Deleting Domain Value Mappings

To delete a selected domain value mapping:

1. Select the domain value mapping to delete.
2. Click **Delete**.

6.3.5 Deleting Domain Value Mapping Tables

To delete the domain value mapping table:

1. Select the domain value mapping table to be deleted and right-click.
2. From the context menu, select **Delete**.
3. Click **YES** in the Confirm Delete dialog.

6.3.6 Modifying Attribute Mappings

To modify a selected attribute mapping, use the Define Mapping page on the Publish Wizard:

See Also: ["Publishing an Event"](#) on page 4-3

1. Select a mapping, and click **Edit**.
2. Edit the appropriate fields.
3. Click **OK**.

6.3.7 Removing Attribute Mappings

To remove attribute mappings, use the Define Mapping page in the Publish Wizard.

- To remove a mapping, delete the attribute and click **Remove**.
- To remove all mappings, click **Clear**.

6.4 Working with Cross-referencing

Creating a cross-reference in iStudio creates a table in the repository schema. To create a cross-reference table:

1. Click **New** from the File menu and select **Cross Reference Tables**. The Create Cross Reference Table dialog is displayed.
2. Enter a name for the cross-reference table in the Table Name field, and click **OK**.

6.4.1 Adding Applications to Cross-reference Tables

To add applications to the cross-reference table:

1. In the Design Navigation list, select the cross-reference table and right-click.
2. From the context menu, select **Add Application**. The Add Application to Cross Reference Table dialog is displayed.
3. From the list, select an application name.
4. Click **OK**.

6.4.2 Removing Applications From Cross-reference Tables

To remove applications from a cross-reference table:

1. In the Design Navigation list, select the cross-reference table and right-click.
2. From the context menu, select **Remove Application**. The Remove Application from Cross Reference Table dialog appears.
3. From the list, select an application name.
4. Click **OK**.

6.4.3 Populating Cross-reference Tables

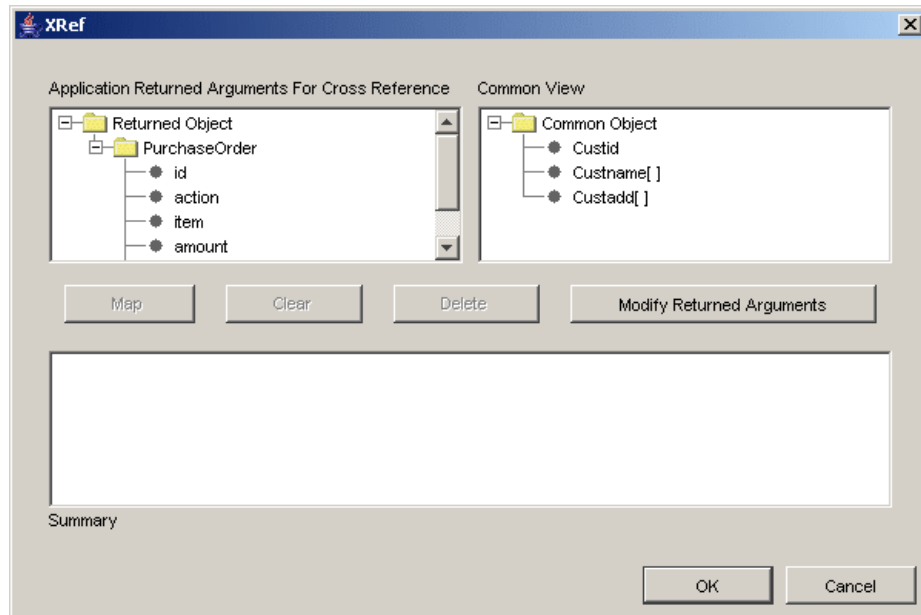
To populate the cross-reference tables, returned arguments must first be defined.

Use the Subscribe Wizard to access the correct page for populating cross-reference tables.

See Also: ["Subscribing to an Event"](#) on page 4-8

To populate cross-reference tables:

1. Click **Cross Reference...** in the Define Application View page. The XRef dialog is displayed.



The Application Returned Arguments box displays the returned arguments. This information is initially populated with the OUT arguments from the application view.

2. Click **Modify Return Arguments** to modify the returned arguments list.
3. Select corresponding attributes in the Application Returned Arguments For XRef and Common View windows, then click **Map**.
4. Specify the Cross-reference Table name to be populated using these attributes values.
5. Click **OK**.

Using Oracle Workflow

This chapter discusses using Oracle Workflow to apply business logic to an integration. It contains the following topics:

- [Oracle Workflow Overview](#)
- [OracleAS Integration InterConnect Integration with Oracle Workflow](#)
- [Using Oracle Workflow with OracleAS Integration InterConnect](#)
- [Model Business Process](#)

7.1 Oracle Workflow Overview

Oracle Workflow is integrated with OracleAS InterConnect. In the context of OracleAS InterConnect, Oracle Workflow is used for business process collaborations across two or more applications. A business process collaboration is defined as the conversation between two or more applications in the context of a business process.

OracleAS InterConnect leverages the robust design time and runtime Oracle Workflow business process definition and execution support to make these business processes explicit and manageable.

Note: Knowledge of Oracle Workflow, its tools, and its Business Event System is required to utilize OracleAS Integration InterConnect with Oracle Workflow for business process collaboration. For more information on Oracle Workflow, refer to *Oracle Workflow Administrator's Guide*.

7.1.1 Oracle Workflow Solves Common Business Problems

The following are some of the common business problems that can be solved using Oracle Workflow.

7.1.1.1 Error Management and Compensating Transactions

If there is an interaction problem between two or more applications, the errors arising from the problem can be centrally managed, and suitable remedial actions can be defined and performed.

Example 7–1 Oracle Workflow, OracleAS Integration InterConnect, and Error Management

Consider a situation where it may be required to keep data of an order entry system synchronized with a backend ERP (Enterprise Resource Planning) system. Assume

that a new purchase order is created in the order entry system and an attempt is made to create a corresponding new purchase order through messaging using OracleAS Integration InterConnect in the backend ERP system. The attempt fails. To deal with this scenario, the integrator can utilize Oracle Workflow to automatically send a compensating message to the order entry system to undo the creation of the purchase order and notify the user who created the order.

In the preceding example, OracleAS Integration InterConnect and Oracle Workflow can be used to model the following for every purchase order that is over \$50,000:

- Send a notification to a named approver and wait for approval.
- If approved, send the message to the ERP system. Otherwise send a message to the order entry system to rollback the order creation.

7.1.1.2 Human Interaction

OracleAS Integration InterConnect adds human interaction to better capture business processes. In the preceding example, OracleAS Integration InterConnect and Oracle Workflow can be used to model the following:

For every purchase order that is over \$50,000, send a notification to a named approver, and wait for approval. If approved, then send the message to the ERP system, otherwise send a message to the order entry system to rollback the order creation.

7.1.1.3 Message Junctions

Fan-in and fan-out of messages can be effectively modeled using OracleAS Integration InterConnect and Oracle Workflow. Fan-in messages involve combining two or more messages into one message. Fan-out messages involve splitting one message into two or more messages.

Example 7-2 Fan-in and Fan-out of Messages

For example, a global organization has a centralized Human Resources ERP application in the United States. Each country has one or more local systems that capture local employee information. If a new employee joins the Japanese branch of this organization, data is entered into a local human resources application and local benefits application. Each entry submits a message for adding this information to the centralized system.

The centralized system needs data from both systems combined and will only commit the data if it was entered successfully in both of the local systems connecting to OracleAS Integration InterConnect. Using Oracle Workflow, this process can be modeled so that OracleAS Integration InterConnect routes messages from both local systems to Oracle Workflow. Oracle Workflow waits until it receives both messages, combines the data, and launches a single message to be delivered by OracleAS Integration InterConnect to the centralized human resources system.

7.1.1.4 Stateful Routing

OracleAS Integration InterConnect provides extensive support for stateless routing through event-based and content-based routing features. Using Oracle Workflow, stateful routing can be accomplished. The decision to route can be based on more than the event or the content of the message.

7.1.1.5 Composite Services

An internal (organization focused) or external (customer/partner focused) service can be built through a well-defined set of business processes involving communication

between two or more applications. For example, a brick-and-mortar retail company wants to provide an on-line procurement service to their customers. Behind the user interface are several business processes controlling communication across several internal applications to deliver a robust, high performance service to the customer.

Note: The ability to define explicit business process collaborations is a feature, not a requirement for completing integrations. It is not necessary to utilize Oracle Workflow for integration if the business process definition is simple enough to be implicitly captured in the messaging through the core functionality in iStudio.

7.2 OracleAS Integration InterConnect Integration with Oracle Workflow

This section describes how OracleAS Integration InterConnect and Oracle Workflow are integrated. It includes the following sections:

- Design Time Tools
- Runtime

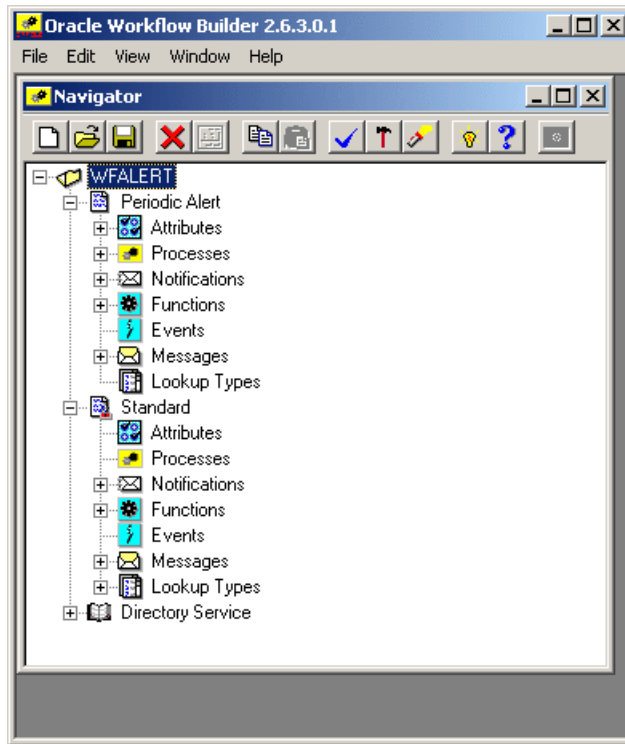
7.2.1 Design Time Tools

During design time, business process and event definitions in iStudio can be deployed to Oracle Workflow. Consequently, Oracle Workflow tools can be launched from within iStudio to graphically create process diagrams in the context of enterprise integration through OracleAS Integration InterConnect.

Using iStudio, the following Oracle Workflow tools can be used:

- Oracle Workflow Builder: Use this tool to complete business process definitions defined and deployed through iStudio. Oracle Workflow Builder creates the process diagrams as shown in [Figure 7-1](#).

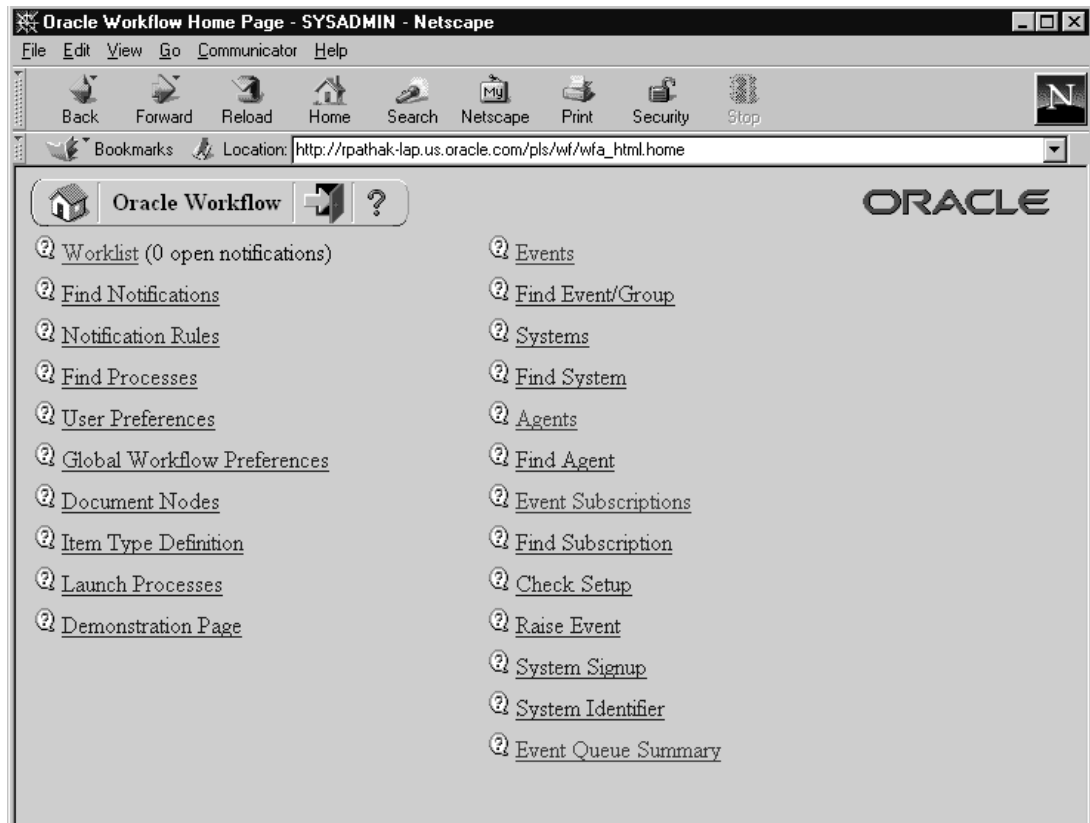
Figure 7-1 Oracle Workflow Builder



- Oracle Workflow Home Page: Use this tool for centralized access to the Web-based features of Oracle Workflow. The Business Event System management and administration is a key feature on this page as shown in [Figure 7-2](#).

See Also: For more information on the Business Event System, refer to *Oracle Workflow Administrator's Guide*.

Figure 7-2 Oracle Workflow Home Page



7.2.2 Runtime

OracleAS Integration InterConnect integrates with the Business Event System of Oracle Workflow. The Business Event System is an application service that uses the Advanced Queuing infrastructure to communicate business events between systems. OracleAS Integration InterConnect registers itself as an external system in Business Event System so the following conditions exist:

- Messages can flow from applications through OracleAS Integration InterConnect, in the common view format, to the Business Event System. The messages will either trigger an event, or continue Oracle Workflow business processes, as defined by iStudio processes and described in Oracle Workflow Builder diagrams.
- Messages can flow from the Business Event System to OracleAS Integration InterConnect in the common view format to applications to either continue or end Oracle Workflow business processes.

At runtime, Oracle Workflow is integrated with OracleAS Integration InterConnect at the hub. Messages are passed between OracleAS Integration InterConnect and the Business Event System of Oracle Workflow using Advanced Queues. The OracleAS Integration InterConnect Oracle Workflow Communication Infrastructure facilitates this communication.

At design time, to keep the integration methodology consistent, iStudio reuses the messaging paradigms of publish/subscribe and request/reply to specify communication between OracleAS Integration InterConnect and Oracle Workflow. For inbound Oracle Workflow messages, an iStudio user can specify using a business process which events Oracle Workflow should subscribe to, and which procedures

Oracle Workflow should implement. For outbound messages, events Oracle Workflow can publish, and procedures it can invoke can be specified.

See Also: ["Using Oracle Workflow with OracleAS Integration InterConnect"](#) on page 7-6

7.3 Using Oracle Workflow with OracleAS Integration InterConnect

OracleAS Integration InterConnect can be used with Oracle Workflow. The following steps describe, in general terms, how to apply business logic:

1. [Model Business Process](#). The user will design the business process using iStudio, and then deploy the process bundles from iStudio to a .wft file. Next, the user will complete the process diagrams in Oracle Workflow Builder using the .wft file.
2. [Deploy Business Processes for Runtime](#). The user will deploy the events to the Business Event System using iStudio. Next, the user will deploy the process diagram to the database using Oracle Workflow Builder.

7.3.1 Model Business Process

To model the business process:

- Design process bundles using iStudio.
- Deploy process bundles from iStudio to a .wft file.
- Complete process diagrams in Oracle Workflow Builder by launching Oracle Workflow Builder from iStudio and using the deployed .wft file.

7.3.2 Deploy Business Processes for Runtime

To deploy business processes for runtime:

- Deploy events to the Business Event System from iStudio.
- Deploy a process diagram from a file to the database using Oracle Workflow Builder.

7.4 Model Business Process

This section describes how iStudio and Oracle Workflow work together in OracleAS Integration InterConnect. It also has instructions on how to use iStudio and Oracle Workflow during design time for business process collaborations across applications. This section includes the following topics:

- [Process Bundle](#)
- [Business Process](#)
- [Activity](#)

7.4.1 Process Bundle

A process bundle is a set of logically-related business processes. The process bundles maps one-to-one with an Oracle Workflow item.

See Also: ["Business Process"](#) on page 7-7

7.4.2 Business Process

A business process contains a set of OracleAS Integration InterConnect common view events or procedures that must be routed through Oracle Workflow in one Oracle Workflow business process. These events and procedures manifest themselves as Oracle Workflow business events and can be used to define a process diagram in Oracle Workflow Builder. It is a one-to-one mapping between OracleAS Integration InterConnect and Oracle Workflow Builder.

7.4.3 Activity

iStudio activities allow the user to define the common view events and procedures that must be a part of an Oracle Workflow business process. The following are types of activities in iStudio:

- **Publish Event:** Oracle Workflow publishes an OracleAS Integration InterConnect common view event. At deployment time, a business event corresponding to the common view event is created in the Business Event system.
- **Subscribe Event:** Oracle Workflow subscribes to an OracleAS Integration InterConnect common view event. At deployment time, a business event corresponding to the common view event is created in the Business Event system.
- **Invoke Procedure:** Oracle Workflow invokes an OracleAS Integration InterConnect common view procedure. At deployment time, two business events corresponding to the common view procedure are created in the Business Event system. One event is for sending the request, and the other is for receiving the reply.
- **Implement Procedure:** Oracle Workflow implements an OracleAS Integration InterConnect common view procedure. At deployment time, two business events corresponding to the common view procedure are created in the Business Event system. One event is for receiving the request, and the other is for sending the reply.

The following table describes how iStudio and Oracle Workflow concepts are mapped.

iStudio Concept	Oracle Workflow Concept	Mapping
Process Bundle	Item	One-to-one
Business Process	Business Process	One-to-one
Common View Event	Business Event	One-to-one ¹
Common View Procedure	Business Event	Two business events per procedure
Publish Activity	Send Event Activity	One-to-one
Subscribe Activity	Receive Event Activity	One-to-one
Invoke Activity	Send Event Activity (for the request) Receive Event Activity (for the reply)	
Implement Activity	Send Event Activity (for the reply) Receive Event Activity (for the request)	

- ¹ Only for events that are part of a business process in iStudio. Events that are part of the common view but not part of a business process are not instantiated as Oracle Workflow business events. All common view events need not be part of business processes. Depending on the integration, some common view events could be exchanged directly between applications without involving Oracle Workflow. These events use the core functionality of OracleAS Integration InterConnect. Other events may need to be part of an explicit business process. This set of events become business events in Oracle Workflow. The same is true for common view procedures.

7.4.4 Creating a Process Bundle

To create a process bundle using iStudio:

1. From the project list, click **Workflow**, and expand the subtree.
2. Right-click **Process Bundles**, and select **New**. The Create Process Bundle dialog is displayed.
3. Enter the name of the process bundle in the Process Bundle Name field.
4. Click **OK**.

7.4.5 Creating a Business Process

To create a business process:

1. From the project list, expand the process bundle for the business process to be created.
2. Right-click **Business Processes**, and select **New**. The Create Business Process dialog is displayed.
3. Enter a name for the business process in the Business Process Name field.
4. Click **OK**.

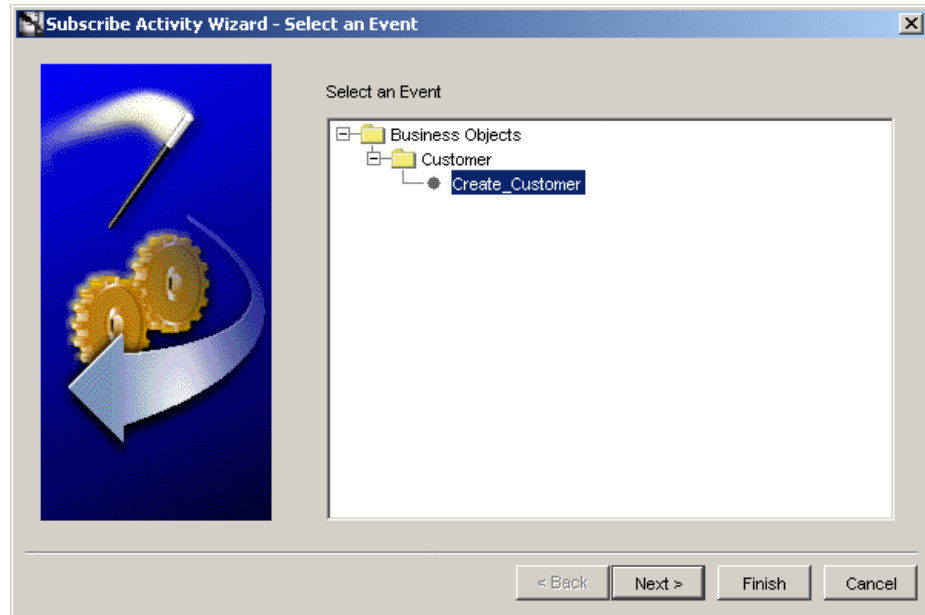
7.4.6 Populating a Business Process with Activities

To populate a business process with activities:

1. From the project list, select the business process to populate.
2. Right-click the business process and in the context menu, select the activity to be part of the business process. Choose from the following activities:
 - **Publish Activity:** Oracle Workflow sends a message to OracleAS Integration InterConnect in the context of the business process.
 - **Subscribe Activity:** Oracle Workflow receives a message from OracleAS Integration InterConnect.
 - **Invoke Activity:** Oracle Workflow sends a request message to OracleAS Integration InterConnect and receives a reply.
 - **Implement Activity:** Oracle Workflow receives a request from OracleAS Integration InterConnect and sends a reply.

The Subscribe Activity Wizard is displayed.

3. Select an event for the activity.



4. Click **Finish**.

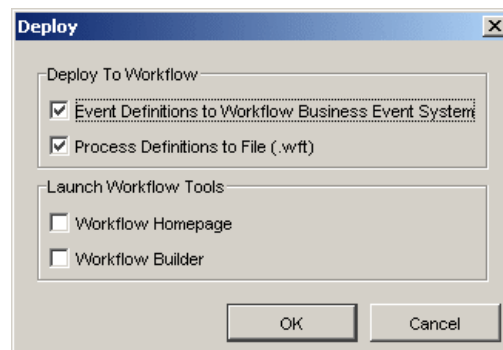
Repeat these steps for adding other activities to the process.

Note: When you create multiple activities under a business process, the list of activities is unordered because the order in which the activities are added is not important. The order can be defined in Oracle Workflow Builder through a process diagram.

7.4.7 Deploying to Oracle Workflow

After populating business processes with activities, the information must be deployed to Oracle Workflow to graphically model a business process. To deploy this information to Oracle Workflow:

1. In the Deploy tab in iStudio, right-click **Workflow**, and select **Deploy To Workflow**. The Deploy dialog is displayed.



2. There are two sets of information that need to be deployed, either independently or together:

- Oracle Workflow Business Events: Business Events need to be created in the Business Event System. This is a requirement for runtime only. You can deploy these after all design-time work, including modeling the process, is complete.

Note: iStudio checks if an event is already deployed before deploying it. You can re-deploy all events at any time. If you deploy an event after all design-time work, then you don't have to re-deploy the event.

To check if events have been deployed, launch the Oracle Workflow Home page.

See Also: "[Launching the Oracle Workflow Home Page](#)" on page 7-12.

- Oracle Workflow Process Definitions through .wft file generation: Information about business processes captured in iStudio provides a foundation for building process diagrams in Oracle Workflow Builder. Deploying process definitions is required for design time.

Note: When deploying process definitions, iStudio prompts for a filename. If an existing file is specified, iStudio will **overwrite** the file. As a result, if there are existing process definitions in a file modified using Oracle Workflow Builder, do not select that filename as the target, otherwise all modifications made will be lost.

By default, both choices are selected. The dialog also allows the following to be automatically launched:

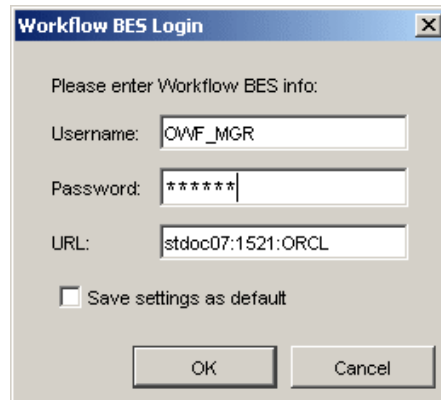
- Oracle Workflow Builder: Defines business process diagrams.
- Oracle Workflow Home Page: Verifies Business Event deployment.

By default, these choices are unselected. Choose to launch these tools with deployment or complete this task at a later time on the Design tab.

See Also: "[Launching Oracle Workflow Tools](#)" on page 7-11

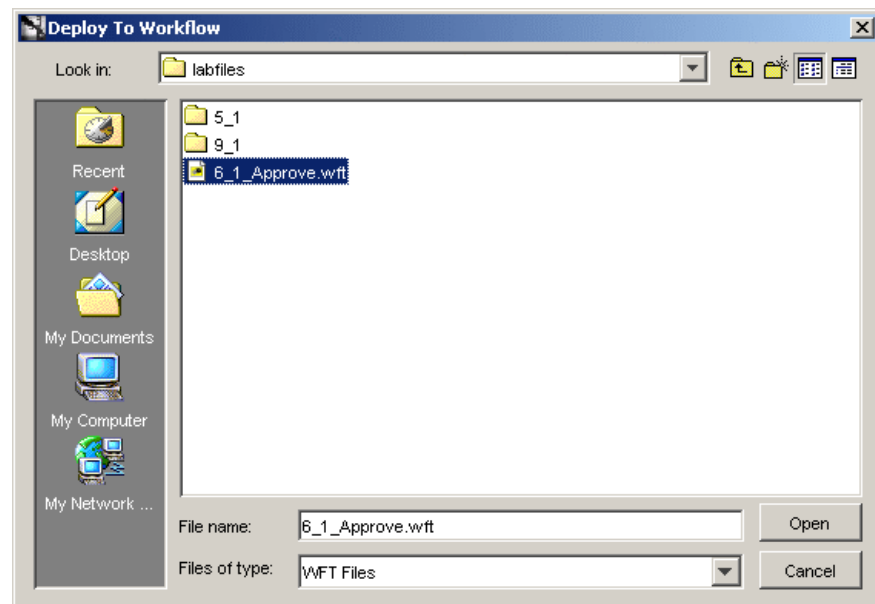
3. Select the appropriate choices, and click **OK**.

If deploying event definitions to the Oracle Workflow Business Event system is selected, the following dialog is displayed.



4. Enter the required information based on the selections made during Oracle Workflow installation, and click **OK**.

If Deploying Process Definitions to a `.wft` file was selected, a file dialog is displayed.



5. Enter the name and location of the file to create, and click **OK**.

Note: When deploying process definitions, iStudio prompts for a filename. If an existing file is specified, iStudio will **overwrite** the file. If there are existing process definitions in a file modified using Oracle Workflow Builder, do not select that filename as the target, otherwise all modifications made will be lost.

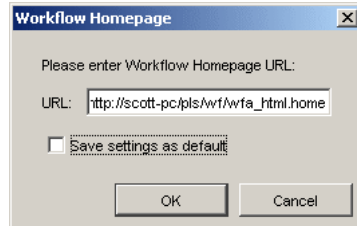
7.4.8 Launching Oracle Workflow Tools

Oracle Workflow tools can be directly accessed through iStudio. You do not need to start Workflow independently. The following sections discuss how to launch Oracle Workflow tools in iStudio.

7.4.8.1 Launching the Oracle Workflow Home Page

To launch the Oracle Workflow Home page:

1. In the Design tab in iStudio, right-click **Workflow**.
2. Select **Launch WF Home Page**. The Workflow Home Page dialog is displayed.

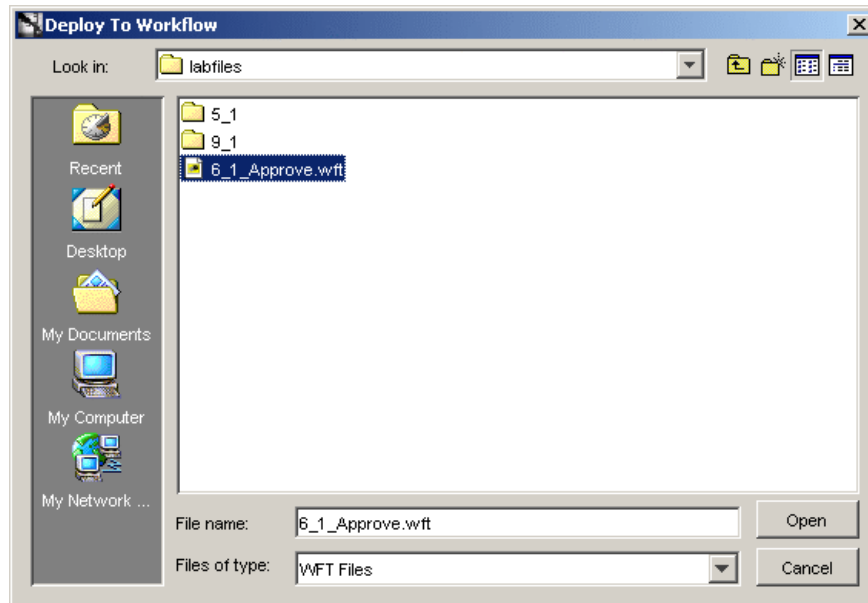


3. Ensure the URL is correct, and click **OK**. The Username and Password Required dialog is displayed.
4. Enter the login information for the Oracle Workflow Home Page, and click **OK**. The Oracle Workflow Home page is launched using the default browser.

7.4.8.2 Launching Oracle Workflow Builder

To launch Oracle Workflow Builder:

1. In the Design tab in iStudio, right-click **Process Bundle** to view in Oracle Workflow Builder.
2. Select **Launch Workflow Builder**. The Deploy To Workflow dialog is displayed.



3. Select an existing `.wft` file name to load into Oracle Workflow Builder. The assumption is that a process definition has already been deployed to a file. Oracle Workflow Builder is launched depending on which process definition file is selected.

Note: To launch Oracle Workflow Builder outside of a specific OracleAS Integration InterConnect process bundle, right-click Workflow and select Launch Workflow Builder.

7.4.9 Modifying Existing Oracle Workflow Processes

When modifying existing Oracle Workflow processes, do not add, modify, or remove OracleAS Integration InterConnect event activities directly in Oracle Workflow Builder. Always make event-related process changes in iStudio, redeploy to the file, and import in Oracle Workflow Builder.

Example 7-3 Oracle Workflow Processes

For example, the following steps must be carried out to create an integration-related Oracle Workflow business process:

1. Create a process bundle in iStudio and create business processes with some activities.
2. Deploy to the `my_process_bundle.wft` file.
3. Import the file into Oracle Workflow Builder.
4. Make **non-event** modifications to the process in Oracle Workflow Builder, such as adding notifications or decision functions to complete the business process.
5. Save the modified process to `my_process_bundle.wft`.

If two new events need to be added to the business process, use the following guidelines:

1. Using iStudio, make the additions to the particular business process.
2. Deploy to a different file such as `changes_to_my_process_bundle.wft`. Do not deploy to `my_process_bundle.wft` because any non-event modifications made through Oracle Workflow Builder will be lost.
3. Launch Oracle Workflow Builder and import both `my_process_bundle.wft` and `changes_to_my_process_bundle.wft`.
4. Move the required modifications from the process representing `changes_to_my_process_bundle.wft` to the process representing `my_process_bundle.wft`.
5. Save the modified process to `my_process_bundle.wft`.

The `my_process_bundle.wft` file now contains the updated process definition.

This chapter describes the deployment tasks in iStudio. It contains the following topics:

- [Deploying PL/SQL Stored Procedures](#)
- [Specifying Application Queue Names for AQ Adapter](#)
- [Deploying Workflow Events and Process Definitions](#)
- [Sync Adapters from iStudio](#)

8.1 Deploying PL/SQL Stored Procedures

iStudio generates PL/SQL stored procedures if the database adapter or the Oracle Applications adapter (only for tables, views, or PL/SQL APIs as interfaces) is used to connect to an application. These stored procedures enable an application to interface with OracleAS Integration InterConnect through the Oracle database. Refer to the respective adapter guides for more information on the content of these stored procedures. This code is generated regardless of the integration point used, which is the event for publish/subscribe or procedure for request/reply, and must be deployed in the application schema to be performed at runtime. There are two ways to deploy this generated PL/SQL code:

- [Manual Deployment](#)
- [Auto Deployment](#)

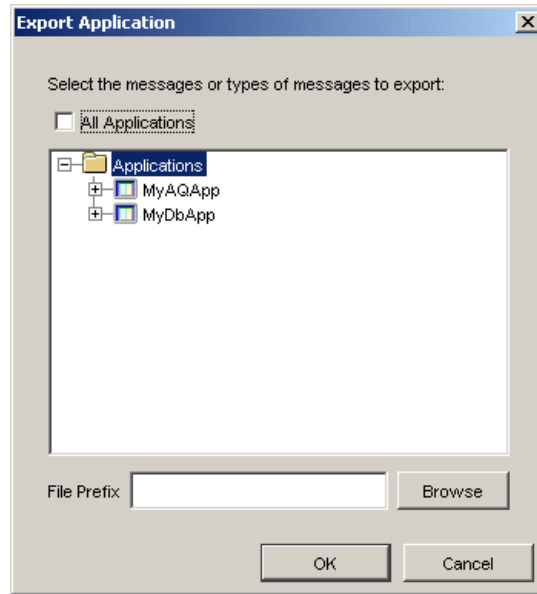
8.1.1 Manual Deployment

This model requires two steps to deploy your code:

- Export the generated code to a file
- Load the exported file manually to the target application schema

This model enables a user to utilize their favorite development environment to modify (if application is receiving data) or build upon (if application is sending data) the generated PL/SQL code. The file export mechanism is also useful for storing the final PL/SQL code in the user desired source control system. To export stored procedures:

1. In the iStudio window, click the **Deploy** tab.
2. Right-click **Applications**, and select **Export PL/SQL**. The Export Application dialog is displayed.



3. 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.
4. 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**.
5. Click **OK**. The stored procedure is now exported to a text file, which is stored in the user specified directory (iStudio directory by default), on your computer.
6. Load the exported file into the target schema. The exported PL/SQL file is deployed for the selected application.

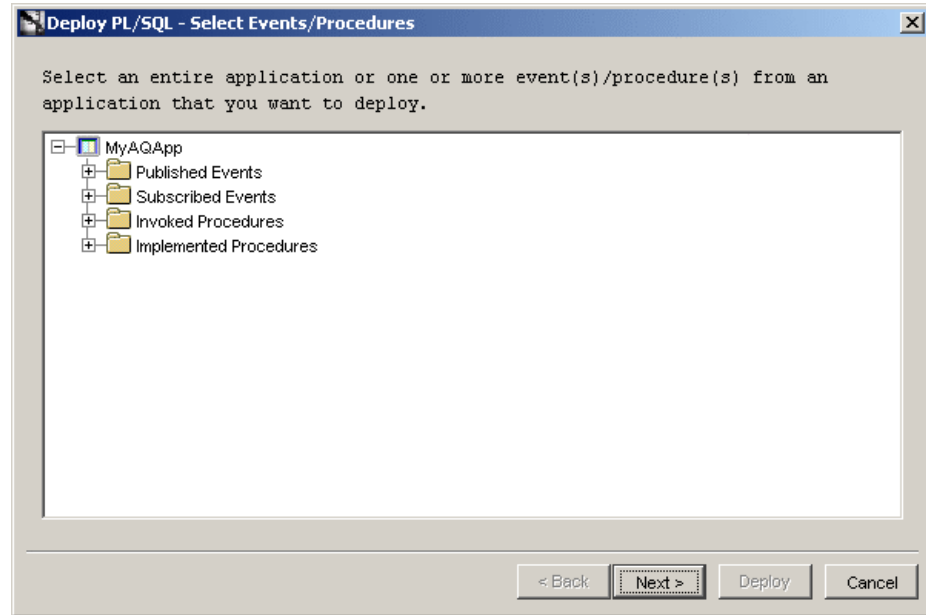
8.1.2 Auto Deployment

This option allows you to deploy the PL/SQL from iStudio, using the Deploy PL/SQL Wizard.

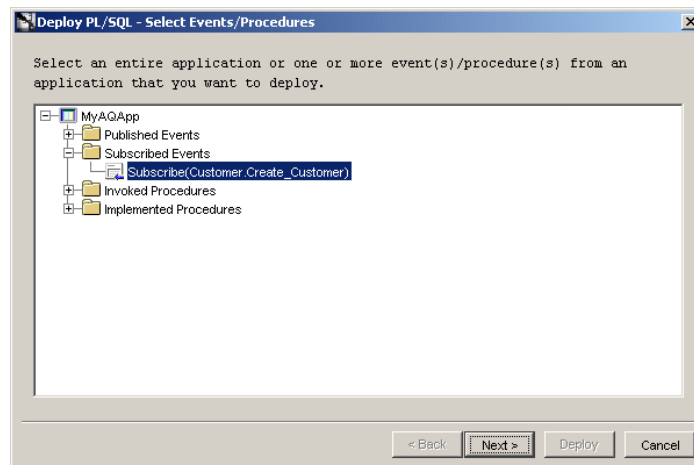
To deploy PL/SQL stored procedures from iStudio:

1. Click the **Deploy** tab in the iStudio window.
2. Right-click a Database application and select **Deploy PL/SQL**. The Deploy PL/SQL - Select Events/Procedures screen is displayed.

The application list displays the published/subscribed events and invoked/implemented procedures.

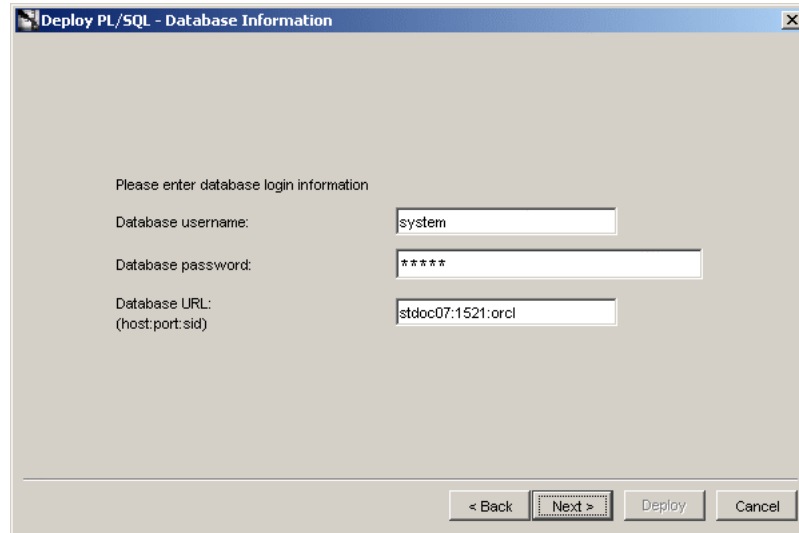


3. Select the application, event or procedure to deploy the corresponding PL/SQL.

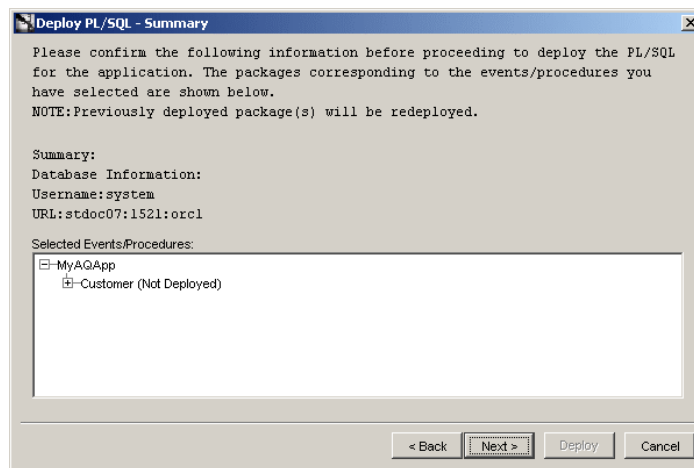


Note: You can deploy PL/SQL code either for one application at a time or at a message level.

4. Click **Next**. The Deploy PL/SQL - Database Information screen is displayed. This page allows you to specify the database connection information for deploying the PL/SQL code.



5. Enter information in the following fields:
 - Database username: The database username required for connecting to the database.
 - Database password: The password required for connecting to the database.
 - Database URL: The URL of the database required for connecting to the database. The URL should be in the form: host:port:SID.
6. Click **Next**. The Deploy PL/SQL: Summary screen appears, which displays a summary of the database connectivity information entered in the previous screen.



7. The Deploy PL/SQL - Summary screen displays the following:
 - Database Information
 - Selected Events/Procedures

This page displays a list of selected packages and the corresponding procedures contained in those packages that you have selected for deployment. The status of each package appears in parenthesis next to the package name.

Note: The status can be any of the following:

Not deployed

Package Status Unknown

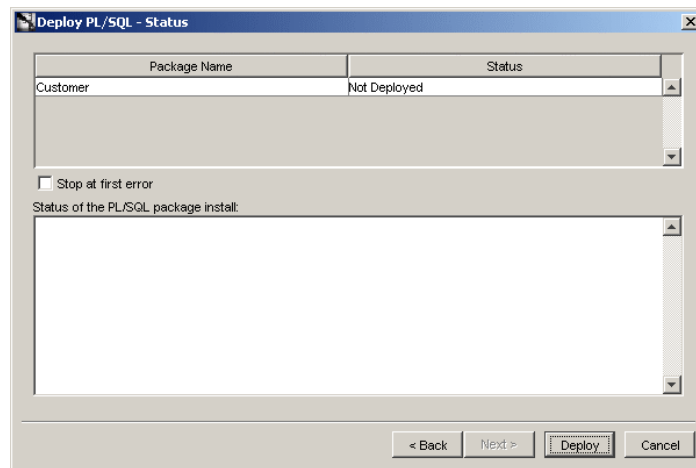
Previously Deployed

Previously Deployed, but package specification invalid

Previously Deployed, but package body invalid

Previously Deployed, but package specification and body invalid

8. Click **Next**. The Deploy PL/SQL - Status screen is displayed.



The Status screen displays:

- A non-editable table: A table that displays the packages of the corresponding application, event or procedure selected for deployment, and the current status of the package.
 - A Stop At First Error checkbox: If this option is checked and multiple packages are being deployed, then the deployment process will stop after encountering the first error. You will be prompted whether you want to continue with the deployment of the remaining packages or not.
9. Click **Deploy**. The generated PL/SQL is deployed for the selected application, event or procedure.

After the deployment is complete, the Status of The PL/SQL Package Install textarea displays the status of the PL/SQL deployment process. If an exception is thrown while executing a PL/SQL stored procedure, it is displayed in this textarea. After deploying each package, the Status column is updated with success or error messages. The messages are:

- Deployed successfully: The PL/SQL package was deployed successfully.
- Deployed, but package specification invalid : The PL/SQL package was deployed. However, the status of the package specification in the database is invalid.
- Deployed, but package body invalid : The PL/SQL package was deployed. However, the status of the package body in the database is invalid.

- Deployed, but package specification and body invalid : The PL/SQL package was deployed. However, the status of both, the package specification and body in the database is invalid.
- Failed to Deploy : The PL/SQL package could not be deployed in the database.
- Package status unknown : An error occurred while deploying the PL/SQL package in the database.

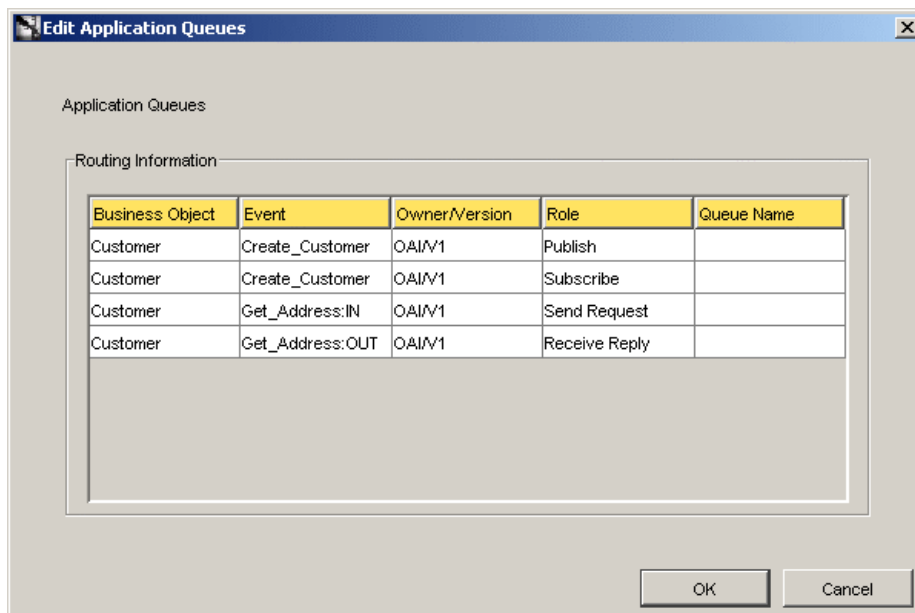
8.2 Specifying Application Queue Names for AQ Adapter

If an application is configured to use the AQ adapter, then the user must specify which queues will be used to send and received data to/from the application. These queues are called application queues.

Note: Application queues are not the same as the hub queues. The hub queue names come preconfigured and all adapters use the hub queues to communicate with each other. Application queues, on the other hand, are used only by applications that expose data to the adapter using AQ. Application queues are peculiar to an AQ adapter enabled spoke system, and the queue names must be specified at deployment time so that the adapter can communicate to the application.

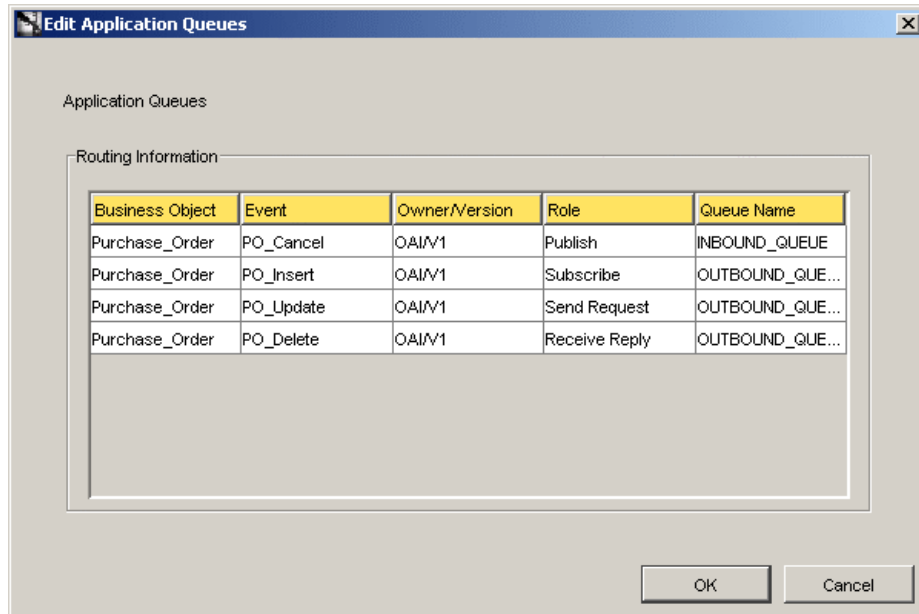
The following steps describe this task.

1. On the Deploy tab in iStudio, expand the Applications list and navigate to AQAPP.
2. Expand the AQAPP node and navigate to the Routing node.
3. Expand the Routing node and select **Application Queues**. The Application Queues property sheet displays on the right side of the iStudio window.
4. Select **Edit** from the Edit menu. This will launch the Edit Application Queues dialog.



5. Add the application Queue name to each event.

Queue Name	Event
INBOUND_QUEUE	PO_Cancel
OUTBOUND_QUEUE	PO_Insert, PO_Update, and PO_Delete



6. Click OK.

8.3 Deploying Workflow Events and Process Definitions

After business processes have been populated with activities, you must deploy the information to Oracle Workflow and then graphically model a business process.

See Also: [Section 7.4.7, "Deploying to Oracle Workflow"](#)

8.4 Sync Adapters from iStudio

Adapters can be configured to cache metadata locally to minimize communication with the repository at runtime. If so configured, these adapters are not aware of changes made through iStudio after they have cached the metadata. Syncing metadata is an explicit way to refresh the adapter local cache with the new repository metadata. The following steps describe this task:

- Select File from the menu bar, then **Sync Adapters**. The Sync Adapters dialog is displayed.
- Select the applications to which to sync adapters, and click **OK**.

Runtime System Concepts and Components

This chapter describes the runtime concepts of OracleAS Integration InterConnect. It contains the following topics:

- [Integration Architecture](#)
- [Components](#)
- [Runtime System Features](#)
- [Real Application Clusters Configuration](#)

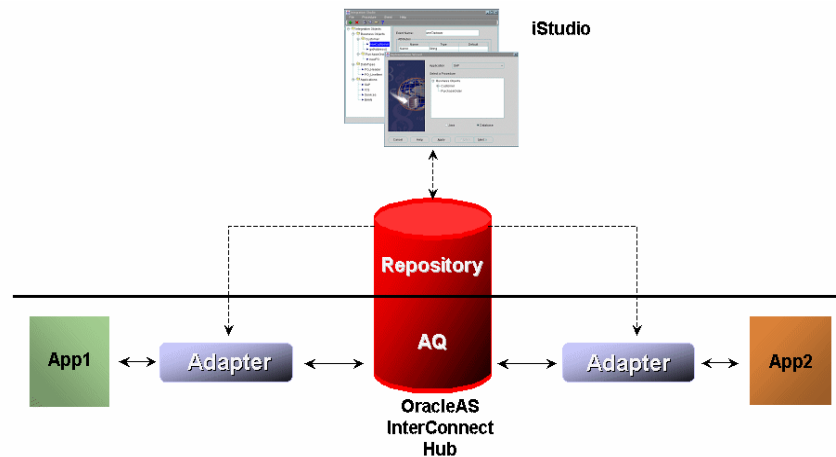
9.1 Integration Architecture

OracleAS Integration InterConnect runtime system is an event-based distributed messaging system. An event is any action that initiates communication through messaging between two or more applications integrated through OracleAS Integration InterConnect. The messaging system can be deployed both within an enterprise or across enterprise boundaries.

The runtime enables inter-application communication through hub and spoke integration. This methodology keeps the applications decoupled from each other by integrating them to a central hub rather than to each other directly. The applications are at the spokes of this arrangement and are unaware of the other applications they are integrating with. To them, the target of a message (or the source) is the hub. As each application integrates with the hub, transformation of data between the application and hub (in either direction) is sufficient to integrate two or more applications.

[Figure 9-1](#) provides an overview of design time and runtime phases in integration.

Figure 9–1 A Graphical Overview of Design Time and Runtime Phases in Integration



9.2 Components

The following are the main components in the runtime system:

- [Adapters](#)
- [Repository](#)
- [Advanced Queues](#)
- [Oracle Workflow](#)

9.2.1 Adapters

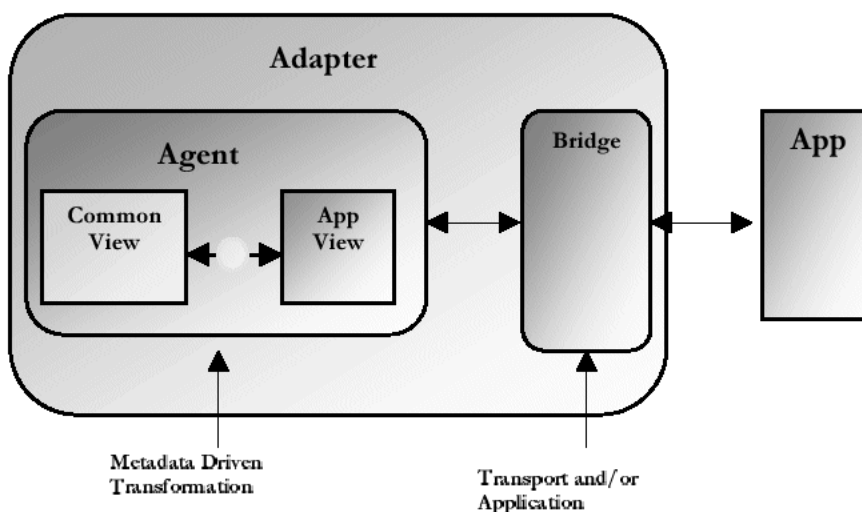
Prepackaged adapters help applications at runtime to participate in the integration without any programming effort.

9.2.1.1 Agent and Bridge Combination

Adapters are the runtime component for OracleAS Integration InterConnect. Adapters have the following features:

- **Application Connectivity:** Connect to applications to transfer data between the application and OracleAS Integration InterConnect. The logical subcomponent within an adapter that handles this connectivity is called a bridge. This protocol/application-specific subcomponent of the adapter knows how to communicate with the application. For example, the database adapter is capable of connecting to an Oracle database using JDBC and calling SQL APIs. This subcomponent does not know which APIs to call, only how to call them.
- **Transformations:** Transform data to and from the application view to common view as dictated by the repository metadata. In general, adapters are responsible for carrying out all the runtime instructions captured through iStudio as metadata in the repository. Transformations are an important subset of these instructions. The logical sub component within an adapter that handles the runtime instructions is called an agent. This is the generic runtime engine in the adapter that is independent of the application to which the adapter connects. It focuses on the integration scenario based on the integration metadata in the repository. There is no integration logic coded into the adapter itself. All integration logic is stored in the repository. The repository contains the metadata that drives this sub

component. For example, in a database adapter, the agent subcomponent knows which SQL APIs to call, but not how to call them. All adapters have the same agent code. It is the difference in metadata that each adapter receives from the repository that controls and differentiates the behavior of each adapter.



Adapters can be configured to cache the metadata at runtime to address performance needs. There are three settings for caching metadata:

- No Caching: For each message, the adapter will query the repository for metadata. This setting is recommended for an early or unstable integration development environment.
- Demand Caching: The adapter will query the repository only once for each message type and then cache that information. For subsequent messages of the same type, it will use the information from the cache. This setting is recommended for a stable integration development environment.
- Full Caching: At start-up time, the adapter will cache all its relevant metadata. This setting is recommended for a production environment.

Note: For more information on the adapters provided by OracleAS Integration InterConnect, refer to *Oracle Application Server InterConnect Installation Guide*.

Adapters are stateless by default. As a result, in case an adapter goes down, the message is either with the application or in the OracleAS Integration InterConnect Hub AQ. This behavior lends itself well to load balancing and high availability requirements for the adapter.

9.2.2 Repository

The repository consists of two components:

- Repository Server: A Java application that runs outside the database. It provides RMI services to create, modify, or delete metadata at design time using iStudio and query during runtime using adapters. Both adapters and iStudio act as RMI clients to communicate with the repository server.

- **Repository Database:** The repository server stores metadata in database tables. The server communicates to the database using JDBC.

Adapters have the ability to cache metadata. If the repository metadata is modified after adapters have cached metadata, the relevant adapters can be notified through iStudio's Sync Adapters functionality.

See Also: [Section 8.4, "Sync Adapters from iStudio"](#)

9.2.3 Advanced Queues

Advanced Queues provide the messaging infrastructure for OracleAS Integration InterConnect in the hub. In addition to being the store and forward unit, they provide message retention, auditing, tracking, and guaranteed delivery of messages.

See Also: *Oracle Database Application Developer's Guide* for information on Advanced Queues

9.2.4 Oracle Workflow

Oracle Workflow facilitates integration at the business process level through its Business Event System. OracleAS Integration InterConnect and Oracle Workflow are integrated to leverage this facility for business process collaborations across applications.

9.3 Runtime System Features

The OracleAS Integration InterConnect runtime features are as follows:

- [Messaging Paradigms](#)
- [Message Delivery](#)
- [Message Retention](#)
- [Routing Support](#)
- [Partitioning](#)
- [High Availability](#)
- [Backup and Recovery](#)

9.3.1 Messaging Paradigms

OracleAS Integration InterConnect runtime supports three major messaging paradigms:

- Publish/Subscribe
- Request/Reply (synchronous and asynchronous)
- Point-to-Point

Point-to-Point messaging can be achieved both in the context of Publish/Subscribe and Request/Reply by using Content Based Routing.

Applications can be configured (per integration point) to support any of these paradigms.

See Also: [Chapter 1, "Getting Started with OracleAS Integration InterConnect"](#)

9.3.2 Message Delivery

The following are features of message delivery:

- **Guaranteed Delivery:** All messages are guaranteed to be delivered from the source applications to the destination applications.
- **Exactly Once Delivery:** The destination applications will receive each sent message exactly once. The messages are never lost or duplicated.
- **In Order Delivery:** The messages are delivered in the exact same order as they were sent. This is applicable only when there is one instance of the adapter running per application serviced.

9.3.3 Message Retention

Messages remain in the runtime system until they are delivered. Advanced Queues in the hub provide the message retention. Messages are deleted when each application that is scheduled to receive a specific message has received that message. For auditing purposes, you can configure the system to retain all successfully delivered messages.

9.3.4 Routing Support

Routing is a function of the Advanced Queues in the hub. By default, `oai_hub_queue` is the only multiconsumer Advanced Queue configured as the persistent store for all messages for all applications. This queue will handle all standard as well as content-based routing needs. The queue is created automatically when you install the repository in the hub. The only reason to change this configuration is if Advanced Queues becomes a performance bottleneck. This is unlikely because most of the message processing is done in the adapters, not in the hub.

See Also: ["Partitioning"](#) on page 9-5

9.3.4.1 Content-Based Routing

Content-based routing allows you to route messages to specific destination applications based on message content. For example, an electronic funds transaction settlement application is designed to transmit bank transactions with a specific bank code to identify the destination bank system. When the electronic funds transfer application publishes a message at runtime, the OracleAS Integration InterConnect runtime component determines the bank code value based on metadata stored in the repository, and routes the message to the corresponding recipient system.

9.3.5 Partitioning

OracleAS Integration InterConnect uses partitioning to manage load balancing across different instances of the same adapter. At runtime, it is possible that the adapter attached to a particular application becomes a performance bottleneck. You can detect this by monitoring the message throughput information using the InterConnect Manager.

OracleAS Integration InterConnect addresses adapter scalability through a well-defined methodology.

Multiple adapters can be attached to one application to share the message load. This can be done in several ways depending upon the needs of your integration environment. For example, Application A publishes three different kinds of events: EventA, EventB, and EventC. Three potential scenarios should be examined to

determine how one or more adapters could be attached to the application to meet performance objectives.

Scenario 1

The order in which the messages are sent by application A must be strictly adhered to for the life of the messages. Messages sent by application A must be received by the subscribing applications in the same order across the different event types.

Recommendation In this case, you cannot add more than one adapter to Application A for load balancing.

Scenario 2

The order in which messages are sent by Application A must be adhered to but not across different event types. Application A publishes the following messages in order: M1_EventA, M2_EventB, M3_EventA. M1_EventA and M3_EventA must be ordered with respect to each other because they correspond to the same event type. M2_EventB has no ordering restrictions with respect to M1_EventA and M3_EventA.

Recommendation In this case, you can leverage the Partitioning feature enabled through iStudio's Deploy tab. This feature allows you to allocate specific adapters for specific message types thereby segmenting the runtime load processing. For this scenario, you can create two partitions: Partition1 corresponds to EventA and Partition2 corresponds to EventB. Dedicate one adapter to each partition (specified at adapter install time or through modification of `adapter.ini` after install). The end result: The order of messages is maintained as per requirements and the processing power has doubled because of two adapter servicing the messages instead of just one. This kind of partitioning is called Message-based partitioning.

Scenario 3

There is no message order dependency, even within the same event type.

Recommendation Two approaches for load balancing are available:

1. One or more adapters are added utilizing the entire Message Capability Matrix. This means that at runtime any one of the adapters would be available to receive any message, though only one of them would actually receive the message. The adapter that is first to request the next message for processing will determine the adapter that will receive the message. This is called Pure Load Balancing partitioning.
2. Message-based Partitions are created based on projections of the number of messages for a particular event type. For example, if there will be three times as many EventA messages than EventB or EventC messages, you could create two partitions: one for handling EventA messages, and the other for handling the other two event types. Now you can dedicate several adapters to handle the EventA message load only. Fewer adapters can be dedicated to the other two event types.

9.3.6 High Availability

Enterprise applications need high availability (HA) because they cannot afford downtime. OracleAS Integration InterConnect uses Oracle Process Manager and Notification (OPMN), Oracle Database Server, and Oracle Real Application Clusters to enable high availability for its components.

See Also: *Oracle Application Server High Availability Guide*

9.3.7 Backup and Recovery

The OracleAS Backup and Recovery feature can be used to back up the critical configuration files for any OracleAS Integration InterConnect 10g Release 2 (10.1.2) installation. You can use the `config_misc_files.inp` file provided by the OracleAS Backup and Recovery tool to back up InterConnect configuration files. The `config_misc_files.inp` file is located in the following directory:

```
$ORACLE_HOME/backup_restore/config
```

See Also: *Oracle Application Server Administrator's Guide*

The following files should be backed up from the OracleAS Integration InterConnect install along with other Application Server component files.

[Hub Component]

```
$ORACLE_HOME/integration/interconnect/hub/hub.ini
$ORACLE_HOME/integration/interconnect/repository/repository.ini
$ORACLE_HOME/integration/interconnect/security/cwallet.sso
$ORACLE_HOME/integration/interconnect/security/ewallet.p12
$ORACLE_HOME/integration/interconnect/adapters/workflow/adapter.ini
$ORACLE_HOME/integration/interconnect/adapters/workflow/ErrorManagement.xml [if
file exists]
```

[Adapter Component]

```
$ORACLE_HOME/integration/interconnect/adapters/<adaptername>/adapter.ini
$ORACLE_HOME/integration/interconnect/adapters/<adaptername>/ErrorManagement.xml
[if file exists]
$ORACLE_HOME/integration/interconnect/security/cwallet.sso [if adapter not
installed in the same midtier as hub]
$ORACLE_HOME/integration/interconnect/security/ewallet.p12 [if adapter not
installed in the same midtier as hub]
```

You can append the preceding mentioned OracleAS Integration InterConnect configuration file names to the `config_misc_files.inp` file with the same file name format.

If all files in a directory have to be backed up, then you can specify only the directory names or use wildcards. You can also exclude certain files from the backup by specifying those file names in the `config_exclude_files.inp` file. However, you cannot specify directories or use wildcards in the `config_exclude_files.inp` file, only single entries are allowed.

9.4 Real Application Clusters Configuration

In Real Application Clusters environment, all active instances can concurrently perform transactions against a shared database. Real Application Clusters coordinates each instance's access to the shared data to provide data consistency and data integrity. It features balanced workloads among the nodes by controlling multiple server connections during period of heavy use and provide persistent, fault tolerant connections between clients and Real Application Clusters database.

See Also: The following documentation for additional information on Real Application Clusters:

- *Oracle10g Real Application Clusters Administration*
- *Oracle Application Server Concepts*

9.4.1 OracleAS Integration InterConnect Adapters Supporting Real Application Clusters

OracleAS Integration InterConnect adapters leverage Real Application Clusters technology, provide consistent and uninterrupted service without having to restart the adapters, if an instance fails, and provide guaranteed message delivery. OracleAS Integration InterConnect adapters connect to the first of the listed available nodes. Nodes are defined in `adapter.ini` and `hub.ini` files.

See Also: OracleAS Integration InterConnect adapters installation documentation for details on `adapter.ini` and `hub.ini` files associated with specific adapters

If one node fails then the database connection is established with the next available node in the `adapter.ini` or `hub.ini` file recursively until a successful connection. Failover is transparent to the user.

The hub connections for all adapters and the spoke connections for Database and Advance Queuing adapters are RAC enabled. From this release, the adapter process is also RAC enabled.

See Also: Section "Support for Oracle Real Application Clusters" in the *Oracle Application Server Application Developer's Guide Advanced Queuing*

9.4.1.1 Adapter Failover Mechanism

In the earlier OracleAS Integration InterConnect releases, the adapters failed over to the next node in the Real Application Clusters environment for any exception. This release changes the adapter failover mechanism. The adapters are designed to failover only when the corresponding node fails. This means that a normal exception will not cause a failover to be triggered. Instead, the adapter will failover only when the node itself fails.

9.4.2 Configuration

The `adapter.ini` and `hub.ini` files must be populated with the information about the host, port, and instance for all the nodes. Additional sets of parameters which specify the number of nodes are also required to be populated. All existing entries remain the same except a new entry for each node is added. [Table 9-1](#) describes the additional sets of parameters which specify the number of nodes required to be populated.

Table 9–1 Additional Parameters for RAC Configuration

File Name	Parameter
hub.ini	host_num_nodes hub_hostx hub_portx hub_instancex: where x varies between 2 and the number of nodes.
adapter.ini for the Advanced Queuing adapter	ab_bridge_num_nodes aq_bridge_host aq_bridge_port aq_bridge_instance
adapter.ini for the Database adapter	db_bridge_num_nodes db_bridge_schema1_hostx db_bridge_schema1_portx db_bridge_schema1_instancex: where x is a value between 2 and the number of nodes.

Sample hub.ini File

The following is a sample hub.ini file.

```

hub_username=ichub
encrypted_hub_password=<encrypted_password> use $ORACLE_
HOME/integration/<version>/bin/encrypt for encryption
hub_use_thin_jdbc=true
hub_host=dlsun1312
hub_instance=iasdb
hub_port=1521
hub_num_nodes=2
hub_host2=vindaloo
hub_instance2=orcl
hub_port2=1521

```

9.4.3 Sample Database Adapter adapter.ini File that Shows the Spoke Database Entry

The following is a sample adapter.ini file for the Database adapter that shows the spoke database entry.

```

db_bridge_schema1_host=dlsun1312
db_bridge_schema1_port=1521
db_bridge_schema1_instance=iasdb
db_bridge_num_nodes=2
db_bridge_schema1_host2=vindaloo
db_bridge_schema1_port2=1521
db_bridge_schema1_instance2=orcl

```

Using InterConnect Manager

InterConnect Manager is a new utility that takes care of both the runtime management and error handling needs of OracleAS Integration InterConnect. This chapter describes the functionality of the utility in detail, in the following topics:

- [Overview of InterConnect Manager](#)
- [Starting InterConnect Manager](#)
- [Using InterConnect Manager](#)
- [Using InterConnect Manager in Silent Mode](#)

10.1 Overview of InterConnect Manager

InterConnect Manager is a command-line based, menu-driven utility that allows you to:

- List messages present in the hub queue and hub error table.
- View details and content of specific messages.
- Export messages from queues to files.
- Import messages to queues and resend edited messages.
- Track messages.
- Start and stop adapters.
- Install and uninstall adapters.
- Create and drop the hub schema in the hub database.
- Import/Export the integration metadata repository to and from a file.
- Adapter Error Management.

10.2 Starting InterConnect Manager

InterConnect Manager can run in two modes:

- Command-line, menu-driven mode
- Silent mode

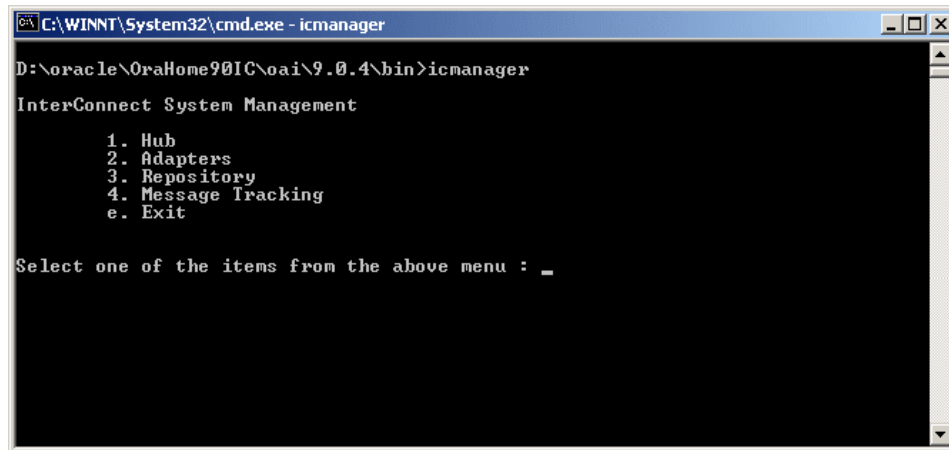
The silent mode is provided for calling the utility from another script.

To start InterConnect Manager in the command-line mode, enter the following command at the prompt:

```
ICManager [-properties hub.ini]
```

InterConnect Manager gets all its information from `OAIHOME/hub/hub.ini`. If you want it to take information from another location instead, use the `properties` parameter and provide the absolute path of the `hub.ini` file as argument. This brings up the main menu of InterConnect Manager as shown in [Figure 10-1](#). From here, you can choose to manage the hub, adapters, and repository. You can also track messages across the OracleAS Integration InterConnect system, from one application to the other.

Figure 10-1 InterConnect Manager Main Menu

A screenshot of a Windows command prompt window titled "C:\WINNT\System32\cmd.exe - icmanager". The prompt shows the command "D:\oracle\OraHome90IC\oai\9.0.4\bin>icmanager" and the output "InterConnect System Management". Below this, a numbered list of options is displayed: "1. Hub", "2. Adapters", "3. Repository", "4. Message Tracking", and "e. Exit". At the bottom, a prompt asks "Select one of the items from the above menu : _" with a cursor under the underscore.

```
C:\WINNT\System32\cmd.exe - icmanager
D:\oracle\OraHome90IC\oai\9.0.4\bin>icmanager
InterConnect System Management
    1. Hub
    2. Adapters
    3. Repository
    4. Message Tracking
    e. Exit

Select one of the items from the above menu : _
```

10.3 Using InterConnect Manager

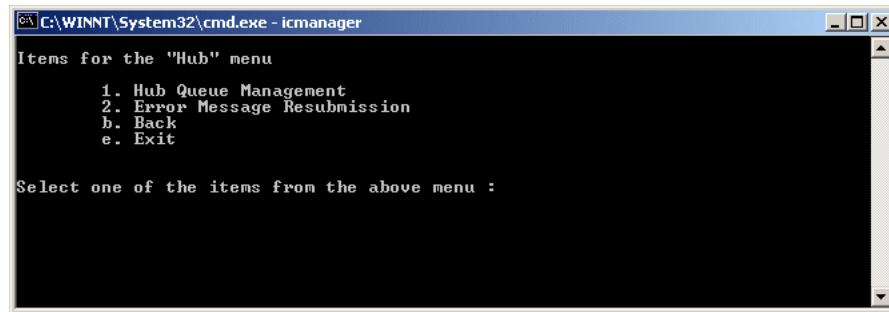
The main menu of InterConnect Manager consists of the following options:

- [Hub](#)
- [Adapters](#)
- [Repository](#)
- [Message Tracking](#)

To select any option, enter its number at the prompt. Each option has further menu options under it.

10.3.1 Hub

When you choose Hub in the main menu, the menu shown in [Figure 10-2](#) is displayed.

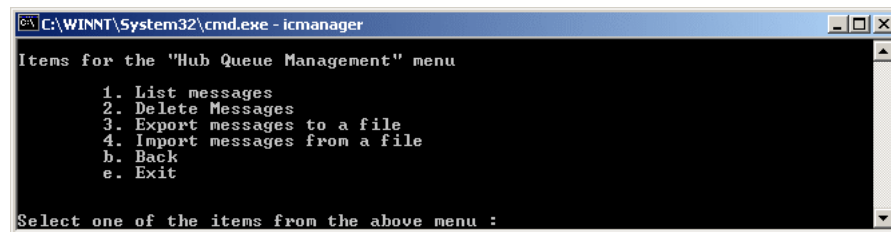
Figure 10–2 Hub Menu

Use the Hub menu to manage the hub queues, and to view and manage the errors that may have occurred during message delivery. Key menu options include:

- [Hub Queue Management](#)
- [Error Message Resubmission](#)

10.3.1.1 Hub Queue Management

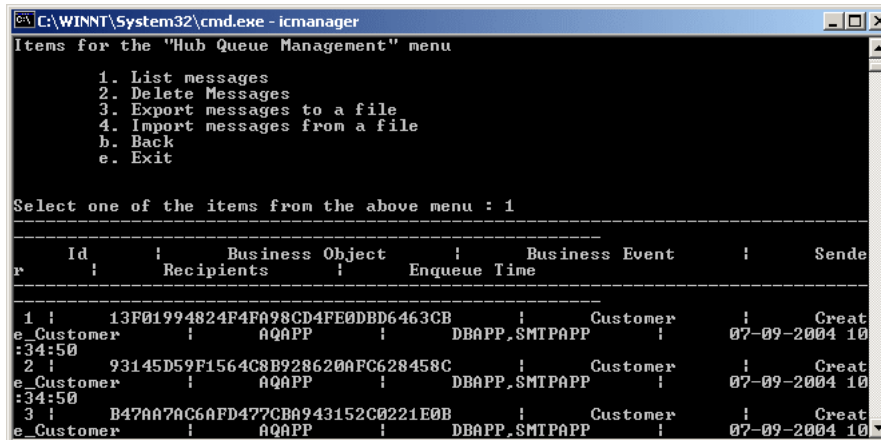
When you choose Hub Queue Management in the Hub menu, the menu shown in [Figure 10–3](#) is displayed.

Figure 10–3 Hub Queue Management menu

Hub Queue Management allows you to view and manage messages present in the hub queue. When messages arrive from adapters, they are placed in the hub queue. The hub processes each message in the queue on a first-come, first-serve basis, applying various routing rules and sending the messages to various adapters. You can also export one or more messages to a file, edit and import them. Key menu options include:

- [List Messages](#)
- [Delete Messages](#)
- [Export Messages To a File](#)
- [Import Messages From a File](#)

10.3.1.1.1 List Messages When you choose the List Messages option in the Hub Queue Management menu, details of all the messages present in the hub queue are displayed as shown in [Figure 10–4](#).

Figure 10–4 List Messages


```

C:\WINNT\System32\cmd.exe - icmanager
Items for the "Hub Queue Management" menu

1. List messages
2. Delete Messages
3. Export messages to a file
4. Import messages from a file
b. Back
e. Exit

Select one of the items from the above menu : 1

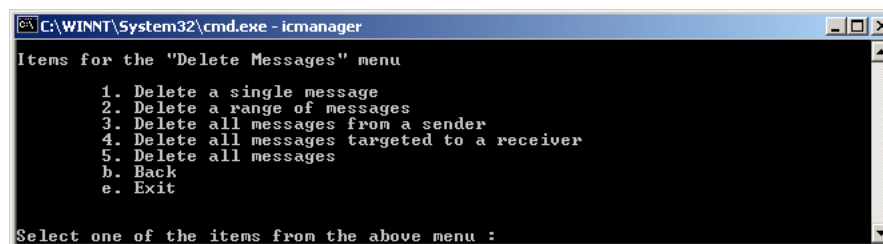
-----
r  Id      Business Object      Business Event      Sender
  |      | Recipients          | Enqueue Time      |
-----
1  |      | 13F01994824F4FA98CD4FE0DBD6463CB | Customer          | Create
e_Customer | AQAPP          | DBAPP,SMTAPP      | 07-09-2004 10
:34:50
2  |      | 93145D59F1564C8B928620AFC628458C | Customer          | Create
e_Customer | AQAPP          | DBAPP,SMTAPP      | 07-09-2004 10
:34:50
3  |      | B47AA7AC6AFD477CBA943152C0221E0B | Customer          | Create
e_Customer | AQAPP          | DBAPP,SMTAPP      | 07-09-2004 10

```

The message details include:

- **Id:** Id acts as a unique identifier for each message in the system. No two messages will have the same Id.
- **Enqueue Time:** Enqueue Time is the time when the hub receives the message from the adapter.
- **Business Object:** The Business Object field contains the name of the message corresponding to the business objects.
- **Event:** Each Business Object consists of one or more events. The Event field contains the name of the Business Event that triggered the message.
- **Sender:** The Sender field contains the name of the application that sent the message.
- **Recipients:** The Recipients field contains the names of the applications that will receive the message.

10.3.1.1.2 Delete Messages When you choose the Delete Messages option in the Hub Queue Management menu, the menu in [Figure 10–5](#) is displayed.

Figure 10–5 Delete Messages


```

C:\WINNT\System32\cmd.exe - icmanager
Items for the "Delete Messages" menu

1. Delete a single message
2. Delete a range of messages
3. Delete all messages from a sender
4. Delete all messages targeted to a receiver
5. Delete all messages
b. Back
e. Exit

Select one of the items from the above menu :

```

The key menu options are:

- [Delete a Single Message](#)
- [Delete a Range of Messages](#)
- [Delete All Messages From a Sender](#)
- [Delete All Messages Targeted To a Receiver](#)

Delete All Messages

When an option is chosen, InterConnect Manager displays a list of messages present in the queue, and prompts the user about which messages should be deleted from the queue.

Delete a Single Message

InterConnect Manager requests the index number of the message to be deleted and then removes the message from the queue.

Delete a Range of Messages

InterConnect Manager requests the low range value and the high range value. It then removes all messages from the hub queue with Ids in the specified range.

Delete All Messages From a Sender

InterConnect Manager requests the name of the sender, and removes all its messages present in the hub queue.

Delete All Messages Targeted To a Receiver

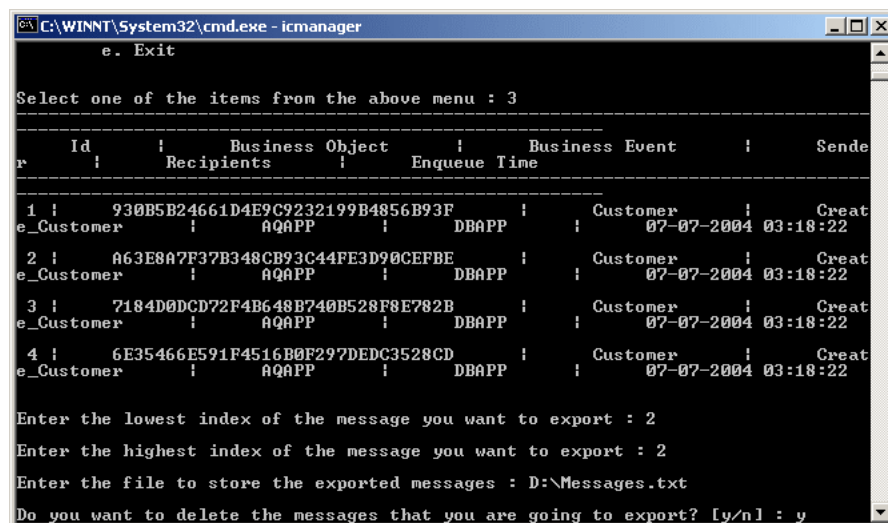
InterConnect Manager requests the name of the receiver, and removes all its messages present in the hub queue. If a message has more than one recipient and one of them is targeted from removal, only the name of the targeted recipient is removed from the message. For example, if you decide to delete all messages targeted to application A, and a particular message in the queue has recipients A and B, the message will not be dropped. Instead, A will be removed from the recipient list.

Delete All Messages

This option removes all messages present in the hub queue.

10.3.1.1.3 Export Messages To a File This option allows you to export a message, or a range of messages to a file. When you choose this option, the menu shown in [Figure 10-6](#) is displayed.

Figure 10-6 Export Messages to a File



```

C:\WINNT\System32\cmd.exe - icmanager
e. Exit

Select one of the items from the above menu : 3
-----
r  Id      | Business Object | Business Event | Sender
  | Recipients | Enqueue Time  |
-----|-----|-----|-----
1 | 930B5B24661D4E9C9232199B4856B93F | Customer | Create
e_Customer | AQAPP | DBAPP | 07-07-2004 03:18:22
2 | A63E8A7F37B348CB93C44FE3D90CEFBE | Customer | Create
e_Customer | AQAPP | DBAPP | 07-07-2004 03:18:22
3 | 7184D0DCD72F4B648B740B528F8E782B | Customer | Create
e_Customer | AQAPP | DBAPP | 07-07-2004 03:18:22
4 | 6E35466E591F4516B0F297DED3528CD | Customer | Create
e_Customer | AQAPP | DBAPP | 07-07-2004 03:18:22

Enter the lowest index of the message you want to export : 2
Enter the highest index of the message you want to export : 2
Enter the file to store the exported messages : D:\Messages.txt
Do you want to delete the messages that you are going to export? [y/n] : y

```

InterConnect Manager displays the list of messages present in the hub queue. It then requests information for the export process. The questions are as follows:

1. Enter the lowest index of the message you want to export:
Enter the lower bound of the range of messages to be exported.
2. Enter the highest index of the message you want to export:
Enter the upper bound of the range of messages to be exported.
If you wish to export only one message, enter the same message Id for both the lower and upper bounds.
3. Enter the file to store the exported messages:
Enter the full path of the text file that stores the exported message. If the file already exists, then it will be overwritten.
4. Do you want to delete the message that you are going to export?
If you wish to drop the messages from the queue, then enter y. Once the messages have been exported to the target file, InterConnect Manager displays a confirmation. You can open the file in any text editor, view the contents, and change them to suit your needs. You can then import the messages back into the hub queue by choosing the Import Messages From a File option in the Hub Queue Management menu.

10.3.1.1.4 Import Messages From a File The Import Messages From a File option allows you to import a message or a range of messages from a file into the hub queue. When you choose this option, the menu shown in [Figure 10-7](#) is displayed.

Figure 10-7 *Import Messages from a File*

```

C:\WINNT\System32\cmd.exe - icmanager
Items for the "Hub Queue Management" menu
  1. List messages
  2. Delete Messages
  3. Export messages to a file
  4. Import messages from a file
  b. Back
  e. Exit

Select one of the items from the above menu : 4
Enter the file to import the messages : D:\Messages.txt
Enter the recipient names separated by comma to send the messages to them : DBAP
P

```

InterConnect Manager requests information for the import process. The questions are as follows:

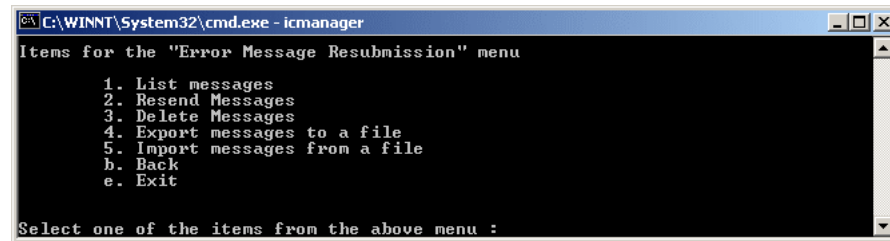
1. Enter the file to import the messages:
The messages that you wish to import must all be present in a single XML file.
Enter the full path of the file that contains the messages to be imported.
2. Enter the recipient name separated by comma to send the message to them:
Enter the names of the application that will receive the imported messages.
Separate the names using commas.

10.3.1.2 Error Message Resubmission

If a message, for some reason, cannot be delivered to the target application by the adapter, it is placed in the Error table of the hub. The Error Message Resubmission option provides you the tools to manage the Error table of the hub, and allows you to carry out various actions on the messages in the queue.

When you choose Error Message Resubmission in the Hub menu, the menu shown in Figure 10–8 is displayed.

Figure 10–8 Error Message Resubmission Menu



Key menu options include:

- [List Messages](#)
- [Resend Messages](#)
- [Delete Messages](#)
- [Export Messages To a File](#)
- [Import Messages From a File](#)

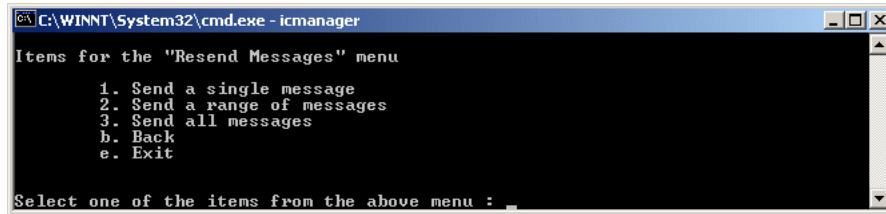
10.3.1.2.1 List Messages The List Messages option provides the details of all the messages present in the error queue. The message details include:

- **Id:** Id acts as a unique identifier for each message in the system. No two messages will ever have the same Id.
- **Enqueue Time:** Enqueue Time is the time when the hub receives the message from the adapter.
- **Sender:** The Sender field contains the name of the application that sent the message.
- **Recipient:** The Recipient field contains the name of the application that was supposed to receive the message.
- **LoggingComponent:** The LoggingComponent field contains the name of the component that logged the error. This helps identify the exact point at which the error occurred.
- **Error Description:** The Error Description field gives a brief description of the error and the action taken.

10.3.1.2.2 Resend Messages The Resend Messages option of the Error Management menu allows you to resend messages that have been put into the Error table. For example, if the message could not be delivered to the target application by the adapter, then the message is moved to the `oai_agent_error` table. But if the adapter is down, then the message will be persisted in the queue, until the adapter is up and running.

When you choose the Resend Messages option in the Error Management menu, the menu shown in [Figure 10-9](#) is displayed.

Figure 10-9 Resend Messages



InterConnect Manager displays another menu where you can choose:

- Send a Single Message
- Send a Range of Messages
- Send All Messages

InterConnect Manager then asks a series of questions related to the resend operation. The questions are as follows:

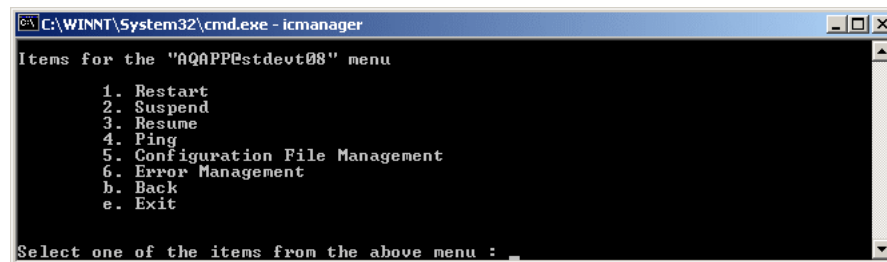
1. Enter the index of the error message you want to resend:
Enter the index number of the message to be resent.
2. Enter the recipient name separated by comma to send the message to them:
Enter the names of the application that will receive the message. Separate the names using commas.
3. Enter the priority for the imported messages:
The priority level decides how quickly the messages will be sent to the recipients. You can choose from level 0-9, with 0 being the lowest and 9 being the highest priority.
4. Do you want to delete the selected error message from the error table?
If you wish to drop the messages from the Error table, then enter y, else enter n.

The functionality of the Delete Messages, Export Messages To a File, and Import Messages From a File options are explained in earlier sections.

See Also: [Delete Messages](#) on page 10-4, [Export Messages To a File](#) on page 10-5, and [Import Messages From a File](#) on page 10-6

10.3.2 Adapters

When you choose Adapters in the main menu, the menu in [Figure 10-10](#) is displayed.

Figure 10–10 Adapters Menu

The complete list of adapters that are present is displayed. When you choose any adapter, InterConnect Manager displays the Adapters menu. Use the Adapters menu to manage the various adapters that form the spokes in the OracleAS Integration InterConnect hub and spoke paradigm. An identical menu is presented for each adapter. Key menu options include:

- Restart: Restarts the adapter.
- Suspend: Temporarily suspends all activity on the adapter. The adapter no longer sends or accepts messages.
- Resume: Allows the adapter to start its normal activities again. This is a counterpart to the Suspend command.
- Ping: Checks to see if the adapter is up and active.
- Configuration File Management
- Error Management

The Configuration File Management and Error Management options are described in detail in the following section.

10.3.2.1 Configuration File Management

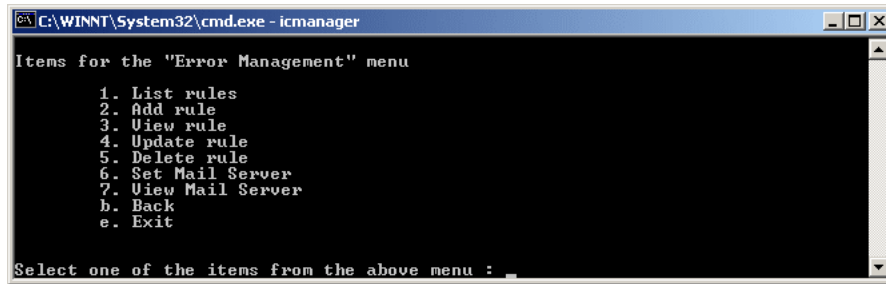
The Configuration File Management option of the Adapter Management menu allows you to manage configuration files for adapters. If you wish to change the behaviour of an adapter, then you must edit its `adapter.ini` file. InterConnect Manager allows you to remotely read and edit the config file.

Key menu options in the Configuration File Management menu include:

- View Config File: Displays the contents of the config file (`adapter.ini` file).
- Edit Config File: Reads the adapter's config file (`adapter.ini` file) from the remote machine and saves it to the local machine.
- Update Config File: Writes the adapter's config file (`adapter.ini` file) to the remote machine where the adapter is installed. You can stop and restart the adapter to reread the config file.

10.3.2.2 Error Management

The Error Management option of the Adapter Management menu allows you to create rules for errors that occur during adapter operation. For example, if an adapter receives messages with an invalid format, the messages are processed in accordance with the rules specified here. [Figure 10–11](#) displays the Error Management Menu.

Figure 10–11 Error Management Menu

Key menu options in the Error Management menu include:

- List Rules
- Add Rule
- View Rule
- Update Rule
- Delete Rule
- Set Mail Server
- View Mail Server

10.3.2.2.1 List Rules The List Rules option displays the list of rules that are currently set for the adapter.

10.3.2.2.2 Add Rule The Add Rule option allows you to add new rules for the adapter error messages.

To add a new rule, enter 2 in the Error Management menu. InterConnect Manager asks a series of questions that help build the new rule for the error messages. The questions are as follows:

1. Enter the name for this rule:

Enter a name for the rule you are about to create. Use alphanumeric characters only and do not use spaces in the name.
2. Enter the error codes separated by comma:

The rule will apply to all error codes listed here.
3. Do you want the adapter to retry the message in case of above errors?

If you enter *y*, then the adapter will retry sending the message. If you enter *n*, then skip to Step 7.
4. How many times would you like to retry the message?

Enter the number of times the adapter must retry sending the message before giving up. If the retries fail, then a message is deleted from the queue.
5. What is the interval for each retry in milliseconds?

Enter the time interval between each retry.
6. Do you want to perform more actions if retry fails?

If you choose *y*, then InterConnect Manager continues with more questions. If you choose *n*, then the questions stop at this point.

7. Do you want to send mail notification?

If you choose *y*, then InterConnect Manager continues with more questions. If you choose *n*, then the questions stop at this point.

8. Enter the From address:

9. Enter the To addresses separated by comma:

10. Enter the subject [\$\$ERROR_CODE and \$\$ERROR_MESSAGE\$\$ can be used as part of the subject]:

You can enter any text in the subject field, which will be used as the subject of the mails sent in case of an error. If you have used the variables \$\$ERROR_CODE\$\$ and \$\$ERROR_MESSAGE\$\$ in the message, then they are dynamically replaced with the appropriate error code and error message before the mail is sent.

11. Enter the message body [\$\$ERROR_CODE and \$\$ERROR_MESSAGE\$\$ can be used as part of the subject]:

You can enter any text in the message body, which will be used as the message sent in case of an error. If you have used the variables \$\$ERROR_CODE\$\$ and \$\$ERROR_MESSAGE\$\$ in the message, then they are dynamically replaced with the appropriate error code and error message before the mail is sent.

10.3.2.2.3 View Rule The View Rule option allows you to view the existing rules.

10.3.2.2.4 Update Rule The Update Rule option allows you to edit the existing rules.

10.3.2.2.5 Delete Rule The Delete Rule option allows you to delete an existing rule.

10.3.2.2.6 Set Mail Server The Set Mail Server option allows you to set the SMTP mail server that will be used to mail updates and errors.

10.3.2.2.7 View Mail Server The View Mail Server option allows you to view the current SMTP mail server.

10.3.3 Repository

When you choose Repository in the main menu, the menu shown in [Figure 10–12](#) is displayed.

Figure 10–12 Repository Menu

```

C:\WINNT\System32\cmd.exe - icmanager
Items for the "Repository" menu
    1. interconnectrepository@stdevt08
    b. Back
    e. Exit

Select one of the items from the above menu : 1
Items for the "interconnectrepository@stdevt08" menu
    1. Restart
    2. Suspend
    3. Resume
    4. Ping
    5. Configuration File Management
    b. Back
    e. Exit

Select one of the items from the above menu :
  
```

Use the Repository menu to manage the hub repository.

Key menu options include:

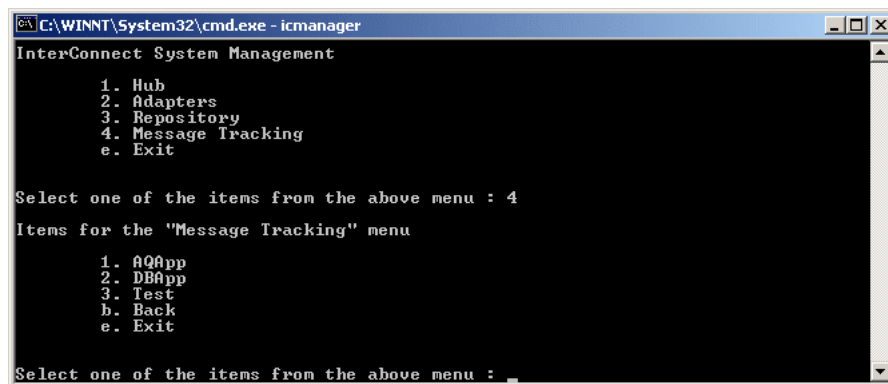
- Restart
- Suspend
- Resume
- Ping
- Configuration File Management

You can carry out all major administrative tasks on the repository from InterConnect Manager. In the case of Configuration File Management option, the functionality is identical to that described on page 10-9, except that the config file read from or written to is the `repository.ini` file.

10.3.4 Message Tracking

When you choose Message Tracking in the main menu, the menu shown in [Figure 10-13](#) is displayed.

Figure 10-13 Message Tracking



The Message Tracking option first lists the applications participating in the integration. When you choose an application, InterConnect Manager lists the Business Objects associated with the application.

When you choose a Business Object, InterConnect Manager lists the Business Events associated with the Business Object.

When you choose a Business Event, InterConnect Manager lists the Message Tracking menu.

Key menu options for the Message Tracking menu include:

- Consolidated Information
- Detailed Information
- Detailed Information For a Tracking ID

When you choose the Consolidated Information option, InterConnect Manager asks you a set of questions that help identify the exact set of messages whose consolidated information needs to be collected and then displays the messages in a tabular format.

When you choose the Detailed Information option, InterConnect Manager directly displays the list of all the messages associated with the particular Business Event.

When you choose the Detailed Information For a Tracking ID option, InterConnect Manager asks for the tracking ID details and then displays the messages in a tabular format.

10.4 Using InterConnect Manager in Silent Mode

InterConnect Manager also has a silent mode of operation, where no menu is displayed and all commands are given directly on the command line. InterConnect Manager commands in the silent mode have the following form:

```
ICManager -component componenttype -name componentname -action actiontype
```

Here, *componenttype* is the type of the component, *componentname* is the name of the component instance that InterConnect Manager should act on, and *actiontype* specifies the action to carry out on the component.

You can use InterConnect Manager to carry out the following activities:

Restart a Component

To restart a component, provide the component type, component instance name and restart as the argument for the action parameter. For example, to restart an adapter called DBApp, the command is:

```
ICManager -component adapter -name DBApp -action restart
```

Similarly, to restart a repository called ICrepo, the command is:

```
ICManager -component repository -name ICrepo -action restart
```

Suspend a Component

To suspend a component, provide the component type, component instance name and suspend as the argument for the action parameter. For example, to suspend the DBApp adapter, the command is:

```
ICManager -component adapter -name DBApp -action suspend
```

Similarly, to suspend the ICrepo repository, the command is:

```
ICManager -component repository -name ICrepo -action suspend
```

Resume a Component

To bring a suspended component into running mode, provide the component type, component instance name and resume as the argument for the action parameter. For example, to resume the DBApp adapter, the command is:

```
ICManager -component adapter -name DBApp -action resume
```

Similarly, to resume the ICrepo repository, the command is:

```
ICManager -component repository -name ICrepo -action resume
```

Check a Component's Availability

To view if a component is running, provide the component type, component instance name and ping as the argument for the action parameter. For example, to view if the DBApp adapter instance is running, the command is:

```
ICManager -component adapter -name DBApp -action ping
```

Similarly, for the ICrepo repository, the command is:

```
ICManager -component repository -name ICrepo -action ping
```

List the Contents of Queues

To view the contents of either the errortable or the hub queue, provide hub as the component type, errortable or queue as the component name and list as the action. For example, to list the contents of the errortable queue, the command is:

```
ICManager -component hub -name errortable -action list
```

Similarly, to list the contents of the hub queue, the command is:

```
ICManager -component hub -name queue -action list
```

Delete the Contents of Queues

To delete the contents of either the errortable or the hub queue, provide hub as the component type, errortable or queue as the component name and delete as the action. For example, to list the contents of the errortable queue, the command is:

```
ICManager -component hub -name errortable -action delete
```

Similarly, to delete the contents of the errortable queue, the command is:

```
ICManager -component hub -name errortable -action delete
```

Integration Scenario

This appendix provides an integration scenario and model based on a fictitious company, Acme, Inc. using OracleAS Integration InterConnect. It contains the following topics:

- [Integration Scenario Overview](#)
- [Modeling the Integration](#)
- [Implementing the Scenario](#)
- [Modeling Business Logic in Oracle Workflow](#)
- [Deployment](#)
- [Conclusion](#)

A.1 Integration Scenario Overview

Each division of Acme, Inc. has multiple Order Fulfillment Systems which are a legacy from various mergers and acquisitions. Maintaining the parts of these systems such as platforms, software, training, and so on is costly and time consuming for Acme. In addition, the lack of integration between the systems prevents business analysis at the enterprise level.

Acme has created a new centralized system, and the first phase of the integration project is to synchronize the purchase order information on one of the legacy systems with the new system.

A.1.1 The New Centralized System

The new order fulfillment system operates on an Oracle10g database and uses the OracleAS Integration InterConnect Database Adapter to communicate with the system.

A.1.2 The Legacy System

The legacy order fulfillment system operates on an Oracle8i database and uses the OracleAS Integration InterConnect Advanced Queuing Adapter to communicate with the system.

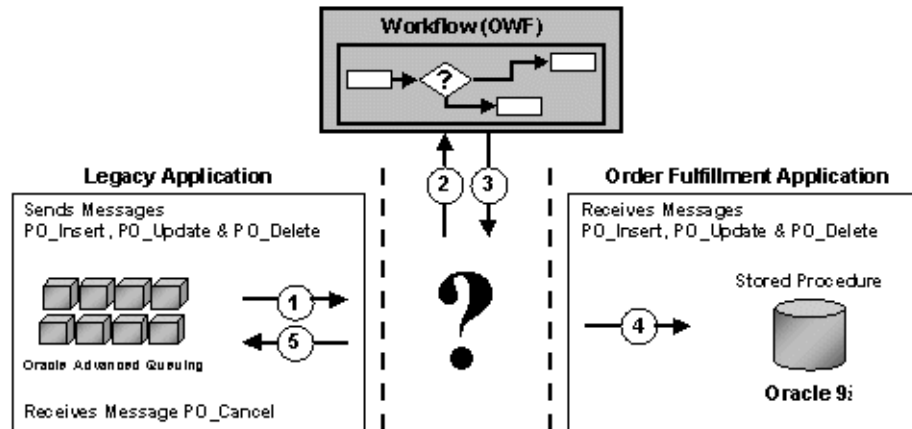
The Purchase Order table in this system has a database trigger to queue the changed records. OracleAS Integration InterConnect is configured to listen to that queue to accomplish the integration.

Note: There are many methods available to capture changes to a system. These methods include, but are not limited to, database triggers, interface tables, and database log files.

A.1.3 The Integration Scenario

Consider an organization that wishes to integrate its legacy system containing its purchase order tables with the new order fulfillment application running on Oracle10g. [Figure A-1](#) illustrates this integration scenario:

Figure A-1 Integration Scenario



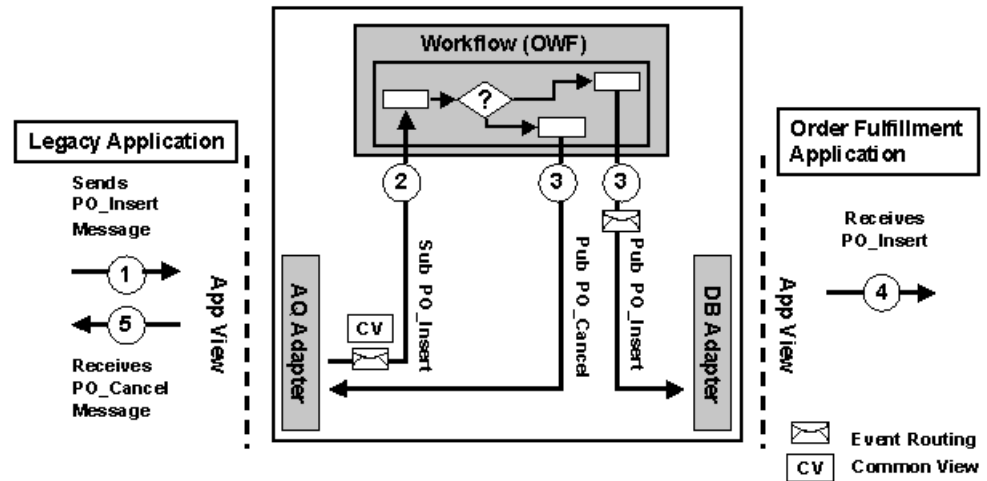
The first step in any integration scenario is to model the integration.

- Legacy System
 - Any change to the Purchase Order table in the legacy application is published using a database trigger. An administrator must approve all changes, such as insert, update, & delete before they are applied to the new order fulfillment System
- Order Fulfillment System
 - If a change is approved, then it is sent to the order fulfillment system. If a change is rejected, then a cancellation notification is sent back the legacy system.
- Additional Issues
 - The process must be non-intrusive. The user cannot alter the structure in either system.
 - Synchronization of the primary keys of each system must be maintained by the integration platform.
 - The integration must be scalable and support addition of systems.

A.2 Modeling the Integration

[Figure A-2](#) illustrates how OracleAS Integration InterConnect integrates with the scenario in [Figure A-1](#).

Figure A-2 Integration Modeling



Now that the integration scenario has been defined:

How are we going to accomplish this task?

- Legacy Application publishes the insert, update, and delete messages to a queue. OracleAS Integration InterConnect Adapter for AQ (Advanced Queuing adapter) is used to send and receive messages to and from the queue.

The Order Fulfillment Application uses a standard Oracle database and the OracleAS Integration InterConnect Adapter for DB (Database adapter).

- All messages are routed to Oracle Workflow to apply user-defined logic.

A.3 Implementing the Scenario

The following sections describe implementing the integration scenario using iStudio.

- Review Legacy System Database Trigger
- Create a Project
- Create the Common View Business Object
- Create Business Object Events
- Create Applications
- Create a Cross Reference Table
- Create Publish Events
- Subscribe to Events
- Create Content-based Routing
- Create an Oracle Workflow Process Bundle
- Deploy the Process Bundle to Oracle Workflow
- Creating Objects in Oracle Workflow for Modeling

A.3.1 Review Legacy System Database Trigger

The source system uses Oracle10g Advanced Queuing to publish changes to the purchase order table. The user creates a database trigger on the purchase order table. When a record is updated, inserted, or deleted and then committed, the trigger enqueues the appropriate payload. The OracleAS Integration InterConnect Advanced Queuing Adapter is configured to listen on this queue.

The following is an example of the code for the database trigger:

```
CREATE OR REPLACE TRIGGER AQAPP.ENQUEUE_PO
    AFTER INSERT OR DELETE OR UPDATE ON AQAPP.PURCHASE_ORDER FOR EACH ROW
DECLARE
    qname                VARCHAR2(20)        := 'OUTBOUND_QUEUE';
    enqueue_options      DBMS_AQ.ENQUEUE_OPTIONS_T;
    message_properties   DBMS_AQ.MESSAGE_PROPERTIES_T;
    msgid                RAW(16);
    recip_agent          SYS.AQ$_AGENT;
    raw_payload          RAW(32767);
    payload              VARCHAR2(256);
BEGIN
    IF INSERTING THEN
        payload := '<?xml version="1.0" standalone="no"?>' ||
            '<PO_Insert>' ||
            '<id>' || :new.id || '</id>' ||
            '<action>' || 'I' || '</action>' ||
            '<item>' || :new.item || '</item>' ||
            '<amount>' || :new.amount || '</amount>' ||
            '<quantity>' || :new.quantity || '</quantity>' || '</PO_Insert>';

    ELSIF DELETING THEN
        payload := '<?xml version="1.0" standalone="no"?>' ||
            '<PO_Delete>' ||
            '<id>' || :old.id || '</id>' ||
            '<action>' || 'D' || '</action>' || '</PO_Delete>';

    ELSIF UPDATING THEN
        payload := '<?xml version="1.0" standalone="no"?>' ||
            '<PO_Update>' ||
            '<id>' || :old.id || '</id>' ||
            '<action>' || 'U' || '</action>' ||
            '<item>' || :new.item || '</item>' ||
            '<amount>' || :new.amount || '</amount>' ||
            '<quantity>' || :new.quantity || '</quantity>' ||
            '<last_updated>' || :new.last_updated || '</last_updated>' || '</PO_Update>';
    END IF;

    raw_payload := UTL_RAW.CAST_TO_RAW( payload );

    DBMS_AQ.ENQUEUE( queue_name        => qname
                    ,enqueue_options    => enqueue_options
                    ,message_properties => message_properties
                    ,payload             => raw_payload
                    ,msgid              => msgid );

EXCEPTION
    WHEN OTHERS THEN NULL;
END;
```

A.3.2 Create a Project

A project is a container for the integration logic pertaining to an integration scenario. The following steps describe creating the PO_Integration project using iStudio.

1. From the **File** menu, select **New Project**. The Create Project dialog is displayed.

2. Enter `PO_Integration` in the Project Name field, and click **OK**. The Repository Information dialog is displayed.
3. Enter the correct repository information, and click **OK**.

See Also: ["Creating a New Project"](#) on page 2-11

A.3.3 Create the Common View Business Object

Each application has its own semantics and syntax. In order to integrate the data from multiple sources, a common view that is semantically compatible is required. The common views are events or procedures that are grouped in a business object, located under the Common Views node in iStudio. In this scenario, all events are grouped under the `Purchase_Order` business object.

The following steps describe creating the `Purchase_Order` business object.

1. From the File menu, select **New**, then select **Business Object**. The Create Business Object dialog is displayed.
2. Enter `Purchase_Order` in the Business Object Name field, and click **OK**.

See Also: ["Creating Business Objects"](#) on page 3-2

A.3.4 Create Business Object Events

In order to integrate data between two or more systems, a semantically compatible view, or common view, is required. In this scenario, the insert, update, delete, and cancel events are grouped under the `Purchase_Order` business object. The following four events must be created:

- `PO_Cancel`
- `PO_Insert`
- `PO_Update`
- `PO_Delete`

Note: When an event is created, a Common Data Type representing its structure is automatically created. This common data type can then be reused to define the structure of other events.

The following steps describe creating the `PO_Insert` event using an XML DTD (Data Type Definition). The user can also use the database or other common data type to describe the structure of the event.

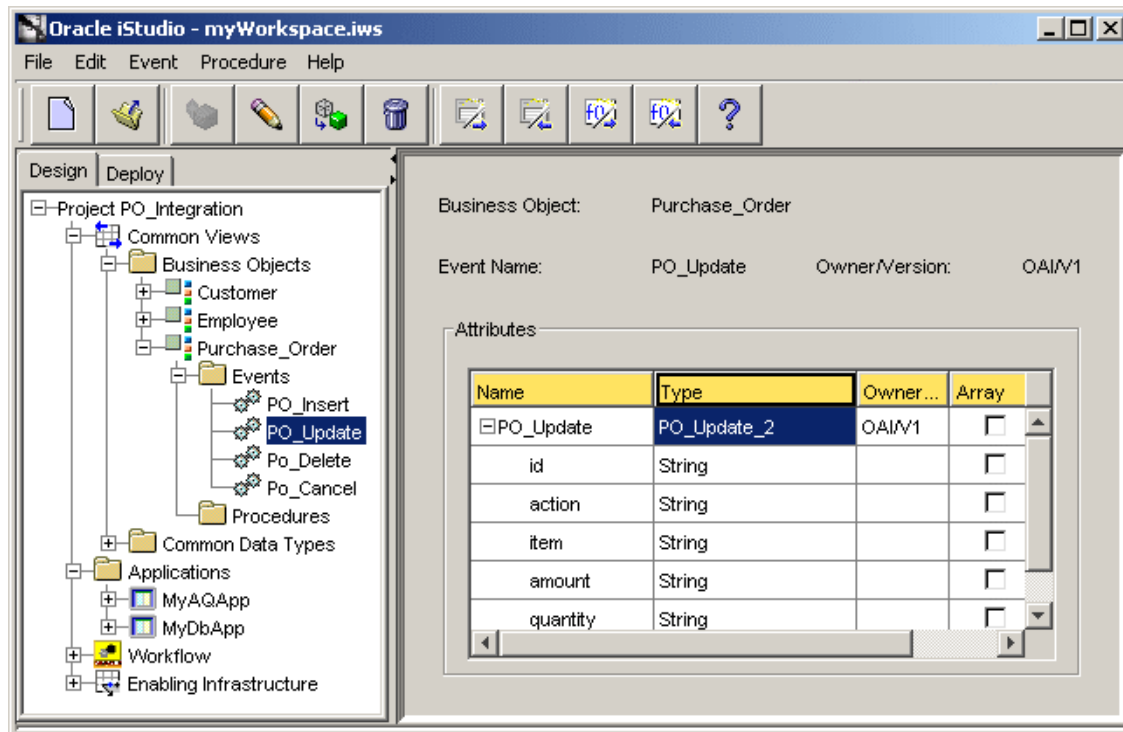
1. From the File menu, click **New**, and then select **Event**. The Create Event dialog is displayed.
2. Select `Purchase_Order` as the Business Object.
3. Enter `PO_Insert` in the Event Name field.
4. Click **Import**, and select **XML**.
5. Select the predefined file, `PO_Insert_CV.dtd` in the Open dialog, and click **Open**.
6. Select `PO_Insert` in the Choose Root Dialog, and click **OK** to return to the Create Event dialog.

7. Click OK.

Use similar steps for the PO_Update, PO_Delete, and PO_Cancel events, substituting the following correct XML DTD for each event. The PO_Cancel, PO_Delete, PO_Insert, and PO_Update events appear in the Design Object Navigator under the Events node as shown in [Figure A-3](#).

See Also: "Creating Events" on page 4-2

Figure A-3 Completed Event Node in iStudio



A.3.4.1 DTD Code

Each event has its own XML DTD. The code for each event is shown.

- PO_Cancel

```
<!ELEMENT PO_Cancel (id, action, item, amount, quantity)>
<!ELEMENT id          (#PCDATA)>
<!ELEMENT action      (#PCDATA)>
<!ELEMENT item        (#PCDATA)>
<!ELEMENT amount      (#PCDATA)>
<!ELEMENT quantity    (#PCDATA)>
```

- PO_Update

```
<!ELEMENT PO_Update (id, action, item, amount, quantity, last_updated)>
<!ELEMENT id          (#PCDATA)>
<!ELEMENT action      (#PCDATA)>
<!ELEMENT item        (#PCDATA)>
<!ELEMENT amount      (#PCDATA)>
<!ELEMENT quantity    (#PCDATA)>
<!ELEMENT last_updated (#PCDATA)>
```

- PO_Delete

```

<!ELEMENT PO_Delete (id, action)>
<!ELEMENT id          (#PCDATA)>
<!ELEMENT action      (#PCDATA)>
■ PO_Insert

<!ELEMENT PO_Insert (id, action, item, amount, quantity)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT action (#PCDATA)>
<!ELEMENT item (#PCDATA)>
<!ELEMENT amount (#PCDATA)>
<!ELEMENT quantity (#PCDATA)>

```

A.3.5 Create Applications

An application in iStudio represents an instance of an adapter communicating with an application. When the user installs an adapter, a unique name is supplied, and in iStudio, this name is used as the name of the application. This scenario demonstrates creating the AQAPP and DBAPP applications.

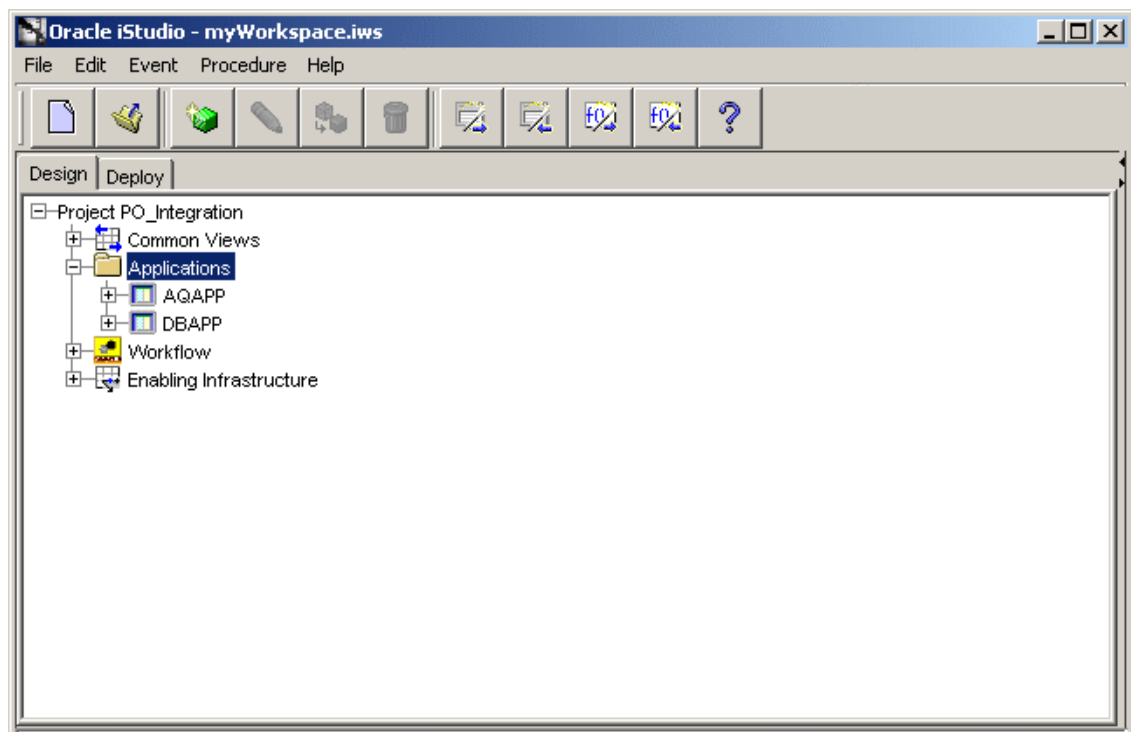
See Also: ["Creating an Application"](#) on page 3-1

The following steps describe creating the AQAPP application using iStudio.

1. From the File menu, select **New**, and then select **Application**. The Create Application dialog is displayed.
2. Enter AQAPP in the Application Name field, and click **OK**.

Complete the same steps to create the DBAPP application. The AQAPP and DBAPP applications appear in the Design Object Navigator under the Applications node as shown in [Figure A-4](#).

Figure A-4 AQAPP and DBAPP Applications in iStudio



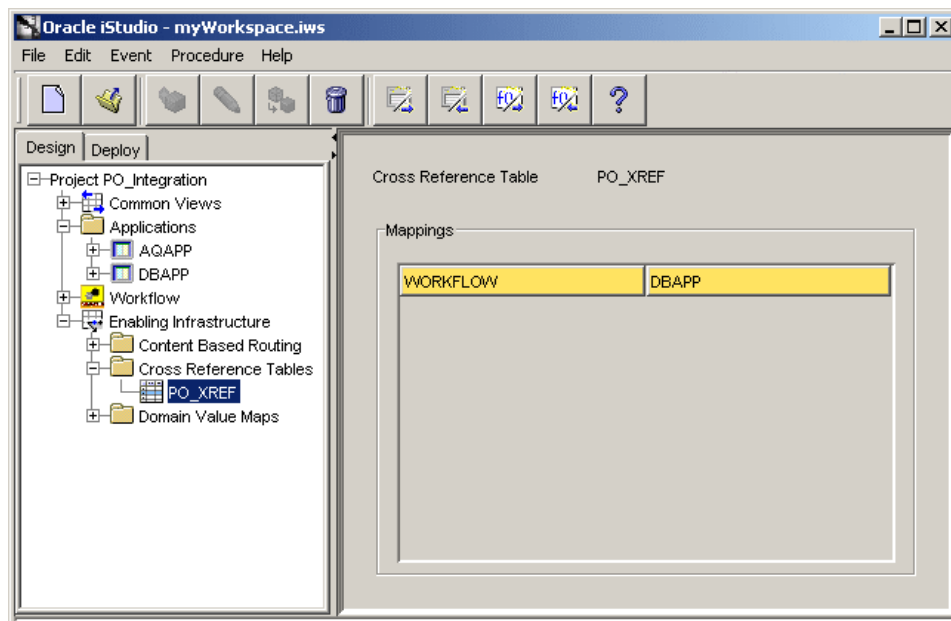
A.3.6 Create a Cross Reference Table

Each system has its own unique identifier or primary key. In most cases, an administrator does not allow any changes to the structure of their systems. As a result, using a cross reference table, the keys of both systems can be maintained and cross-referenced for subsequent updates and deletes.

The following steps describe creating the PO_XREF cross reference table using iStudio. The table is automatically created in the repository schema and is referenced by the subscribing application. The WORKFLOW and DBAPP applications are added to the table, as the publisher and subscriber respectively.

1. From the File menu, click **New**, and then select **Cross Reference Tables**. The Create Cross Reference Table dialog is displayed.
2. Enter PO_XREF in the Table Name field, and click **OK**.
3. Right-click the PO_XREF in the Navigator, and add the WORKFLOW and DBAPP applications. The PO_XREF cross reference table appears in the Design Object Navigator under the Cross Reference Tables node as shown in [Figure A-5](#).

Figure A-5 PO_XREF Cross Reference Table in iStudio



See Also: ["Working with Cross-referencing"](#) on page 6-7

A.3.7 Create Publish Events

The database trigger in the Legacy Application, AQAPP, publishes messages when records are inserted, updated, or deleted in the purchase order table. This process happens outside the OracleAS Integration InterConnect environment. The OracleAS Integration InterConnect Advanced Queuing adapter is configured to read these messages. The publish events under the iStudio application will:

- Map the application view to the common view.
- Perform transformations.

- Publish the new event to subscribers in the OracleAS Integration InterConnect environment.

The following steps describe how the message received from the Legacy Application queue is processed.

Starting the Publish Wizard

To start the Publish Wizard:

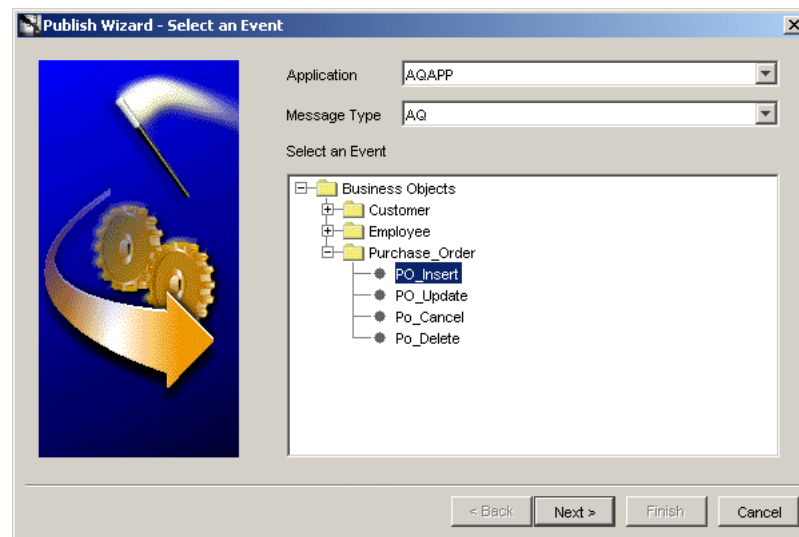
1. Expand the Applications node in the Design Object Navigator.
2. Select and expand the AQAPP application.
3. Select the published events node.
4. Right-click **Published Events**, and select **New**. The Publish Wizard is displayed.

Using the Publish Wizard to Publish the PO_Insert Event

To publish the PO_Insert Event:

1. Select an Event Page
 - a. Enter information in the following fields:
 - * Application: Select AQAPP for the application.
 - * Message Type: Select AQ for the message type.
 - b. Expand the Business Objects list in the Select an Event box and drill down to PO_Insert.
 - c. Select PO_Insert and click **Next**.

Figure A-6 Publish Wizard - Select an Event page



2. Define Application View Page
 - a. Import Attributes

Import attributes from the common view by clicking **Import** and selecting **Common View**. The structure of the PO_Insert common view event is displayed. If the application view is different from the common view, then use the database or an XML DTD to define the structure.

- b. Create an Event Map

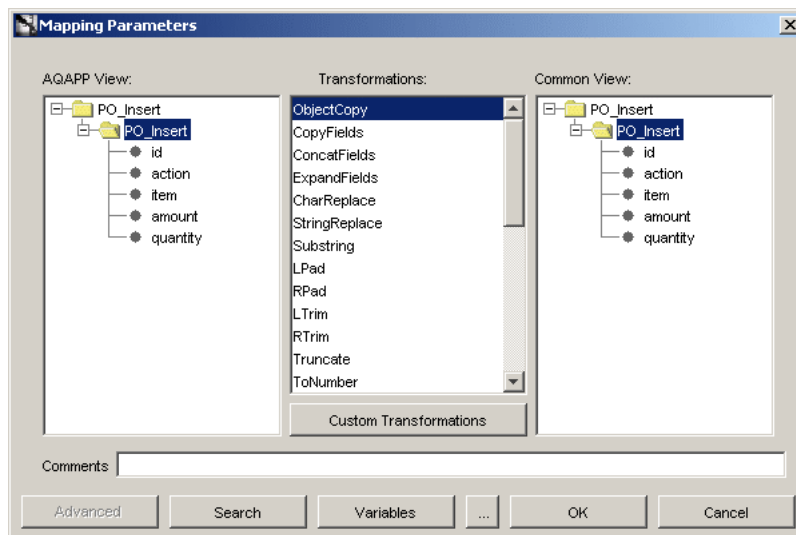
An event is received and converted into a common view. This common view can be mapped by any application. If the structure of one or more events is identical, then routing becomes an issue. An event map is used to distinguish the routing in this situation. The Action field in the application view contains I for insert, U for update, or D for delete. Complete the following steps to create an event map:

 - c. Click **Event Map**, then click **Add**.
 - d. Select the Action field, and enter **I**.
 - e. Click **Add**.
 - f. Click **Next**.
3. Define Mapping Page

Use the Define Mapping page to map fields from the AQAPP View to the common view using transformations. In this scenario the structure is identical, as a result, the `ObjectCopy` transformation is used to map all the fields at once. To define new mappings:

 - a. Click **New**. The Mapping Parameters dialog is displayed.
 - b. Expand the `PO_Insert` list and select the `PO_Insert` node in the AQAPP View box.
 - c. Select **ObjectCopy** in the Transformations box.
 - d. Expand the `PO_Insert` list and select the `PO_Insert` node in the Common View box.
 - e. Click **OK**. The new mapping is displayed in the Summary box of the Define Mapping page.
 - f. Click **Finish**.

Figure A-7 Publish Wizard - Mapping Parameters



To create the `PO_Update` and `PO_Delete` publish events, repeat the same steps, using the following values for steps 2 and 3.

- PO_Update
 - Use the PO_Update common view.
 - The event map value is U.
 - Use the ObjectCopy transformation and map to PO_Update.
- PO_Delete
 - Use the PO_Delete common view.
 - The event map value is D.
 - Use the ObjectCopy transformation and map to PO_Delete.

See Also: ["Publishing an Event"](#) on page 4-3

A.3.8 Subscribe to Events

The DBAPP application subscribes to the following three events:

- PO_Insert
- PO_Update
- PO_Delete

The AQAPP application subscribes only to the PO_Cancel event.

See Also: ["Subscribing to an Event"](#) on page 4-8

A.3.8.1 DBAPP Application Subscriptions

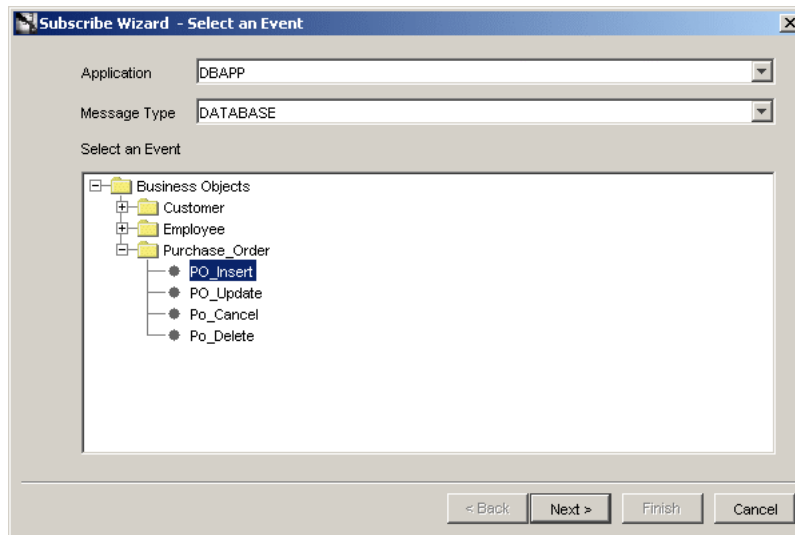
The following steps describe how the Order Fulfillment Application subscribes to messages.

Starting the Subscribe Wizard:

1. In the Design Object navigator, expand the Application node.
2. Select and expand the Application node to display the Subscribed Events node.
3. Right-click **Subscribed Events**, and select **New**. The Subscribe Wizard is displayed as shown in [Figure A-8](#).

Using the Subscribe Wizard to Subscribe to the PO_Insert Event

1. Select an Event Page
 - a. Enter information in the following fields:
 - * Application: Select DBAPP.
 - * Message Type: Select Database.
 - b. Expand the Business Objects node in the Select an Event box and navigate to PO_Insert.
 - c. Select **PO_Insert**, and click **Next**.

Figure A-8 *Subscribe Wizard - Select an Event page*

2. Define Application View Page

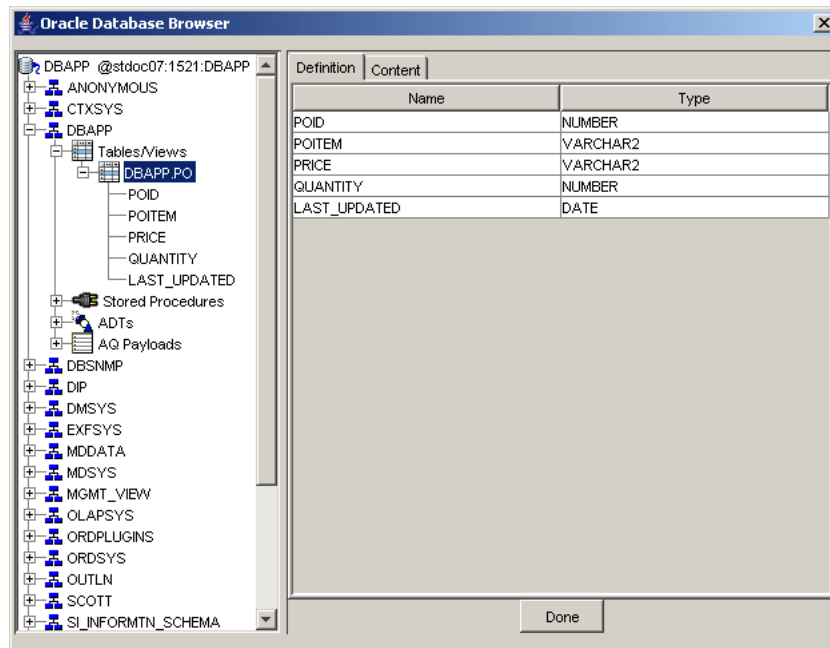
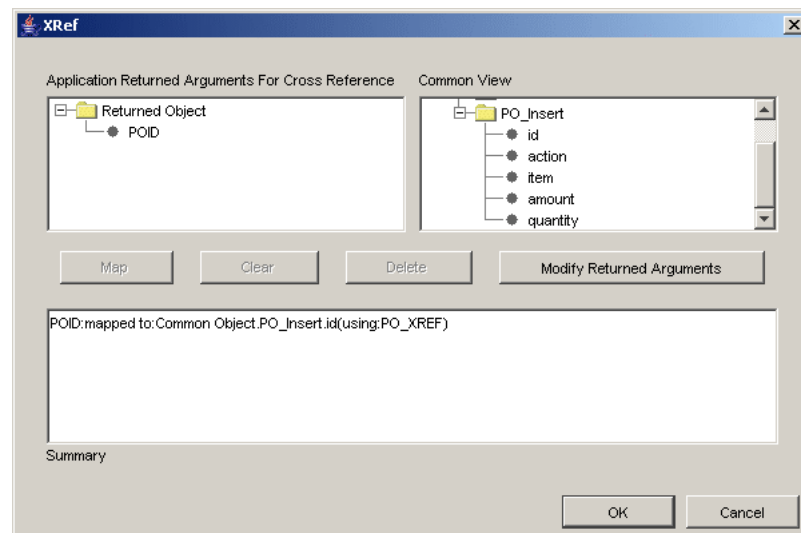
a. Import attributes from the database.

- * Click **Import**, and select **Database**. The Database Login dialog is displayed.
- * Enter the correct information to login to the database, and click **Login**. The Oracle Database Browser dialog is displayed as shown in [Figure A-9](#).
- * In the Browser dialog, expand the Tables/Views node and select DBAPP.PO.
- * Click **Done**.

b. Create a cross-reference.

In "[Create a Cross Reference Table](#)" on page A-8, the PO_XREF cross reference table was created. This table synchronizes the primary keys on the source and target systems.

- * Click **Cross Reference** and select PO_XREF. The XRef dialog is displayed as shown in [Figure A-10](#).
- * Select POID in the Application Returned Arguments For XRef box.
- * Select id in the Common View box.
- * Click **Map**.
- * Click **OK**.

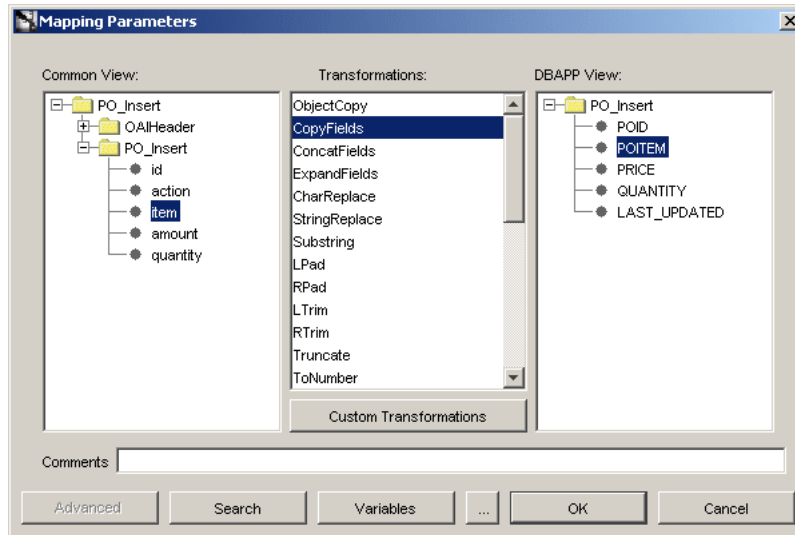
Figure A-9 *Subscribe Wizard - Oracle Database Browser***Figure A-10** *Subscribe Wizard - Cross Reference*

- c. Click **Next**.
3. Define Mapping Page
 - a. Define a new mapping:
 - * Click **New**. The Mapping Parameters dialog is displayed as shown in [Figure A-11](#).
 - * Expand the PO_Insert list and the PO_Insert node in the Common View box. Map the following:

Common View	Transformation	DBAPP View
item	CopyFields	POITEM
amount	CopyFields	PRICE
quantity	CopyFields	QUANTITY

* Click **OK**.

Figure A-11 *Subscribe Wizard - Mapping Parameters*



b. Click **Next**.

4. Define Stored Procedure Page

a. Select sub_PO_Insert_OAI_V1 from the SQL code list. The SQL code is displayed in the box.

b. Add the following code at end of the existing SQL code:

```
PROCEDURE sub_PO_Insert_OAI_V1 ( POID           IN OUT LONG,
                                POITEM         IN LONG,
                                PRICE           IN LONG,
                                QUANTITY        IN NUMBER,
                                LAST_UPDATED    IN DATE)

AS
v_poid NUMBER;

BEGIN
SELECT PO_SEQ.NEXTVAL INTO v_poid FROM dual;
POID :=v_POID;

INSERT INTO PO VALUES
( v_POID, POITEM, PRICE, QUANTITY, SYSDATE );
COMMIT;
END sub_PO_Insert_OAI_V1;
```

c. Click **Finish**.

Create the Subscribed PO_Update Event

To create the subscribed PO_Update Event:

1. Select the PO_Update event.
2. Import the Common View.
3. Define Mapping Page
 - a. Map the same parameters as described in PO_Insert.
 - b. In addition, map the following:
 - * Expand the PO_Update list and node in the Common View box and select id.
 - * Select the LookupXref transformation.
 - * Expand the PO_Update list and select POID in the Application View box.
 - * Click **Apply**. The Mapping dialog is displayed.
 - * Select the Req. checkbox for table listed in the Parameters column and click **OK**.
 - c. Click **Next**.
4. Define Stored Procedure Page

- a. Select sub_PO_Update_OAI_V1 for the SQL code for field. The code is displayed in the box.
- b. Add the following code at end of the existing SQL code:

```
PROCEDURE sub_PO_Update_OAI_V1( POID           IN NUMBER,
                               POITEM        IN LONG,
                               PRICE         IN LONG,
                               QUANTITY      IN NUMBER,
                               LAST_UPDATED  IN DATE)
AS
  v_poid      NUMBER :=poid;
  v_poitem   LONG   :=poitem;
  v_price    LONG   :=price;
  v_quantity NUMBER :=quantity;
BEGIN
  UPDATE PO SET  poitem = v_poitem, price = v_price
                quantity = v_quantity, last_updated = sysdate
  WHERE poid = v_poid;
  COMMIT;

EXCEPTION
  WHEN OTHER THENS NULL;

END sub_PO_Update_OAI_V1;
```

- c. Click **Finish**.

Create the Subscribe PO_Delete Event

To create the subscribe PO_Delete Event:

1. Select the PO_Delete event.
2. Import the Common View.
3. Define Mapping Page

- a. Map the same parameters as described in PO_Insert.
 - b. In addition, map the following:
 - * Expand the PO_Delete list and node in the Common View box and select id.
 - * Select the DeleteXref transformation.
 - * Expand the PO_Delete list and select POID.
 - * Click **Apply**. The Mapping dialog is displayed.
 - * Select PO_XREF from the values column and click **OK**.
 - c. Click **Next**.
4. Define Stored Procedure Page
- a. Select sub_PO_Delete_OAI_V1 for the SQL code for field. The code is displayed in the box.
 - b. Add the following code at the end of the existing SQL code:


```
PROCEDURE sub_PO_Delete_OAI_V1( POID           IN NUMBER,
                                POITEM        IN LONG,
                                PRICE         IN LONG,
                                QUANTITY     IN NUMBER,
                                LAST_UPDATED  IN DATE)

AS
  v_poid      NUMBER :=poid;
BEGIN
  DELETE FROM WHERE PO v_poid = poid;
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN NULL;

END sub_PO_Update_OAI_V1;
```
 - c. Click **Finish**.

A.3.8.2 AQAPP Application Subscriptions

The AQAPP application subscribes to the PO_Cancel event.

1. Select an Event Page
 - a. Enter information in the following fields:
 - * Application: Select AQAPP.
 - * Message Type: Select AQAPP.
 - b. Select PO_Cancel and click **Next**.
2. Define Application View Page
 - a. Import attributes from the common view and click **Next**.
3. Define Mapping Page
 - a. Define a new mapping:
 - * Click **New** and map the following:
Id Copyfields Id
 - * Click **OK**.

- b. Click **Finish**.

A.3.9 Create Content-based Routing

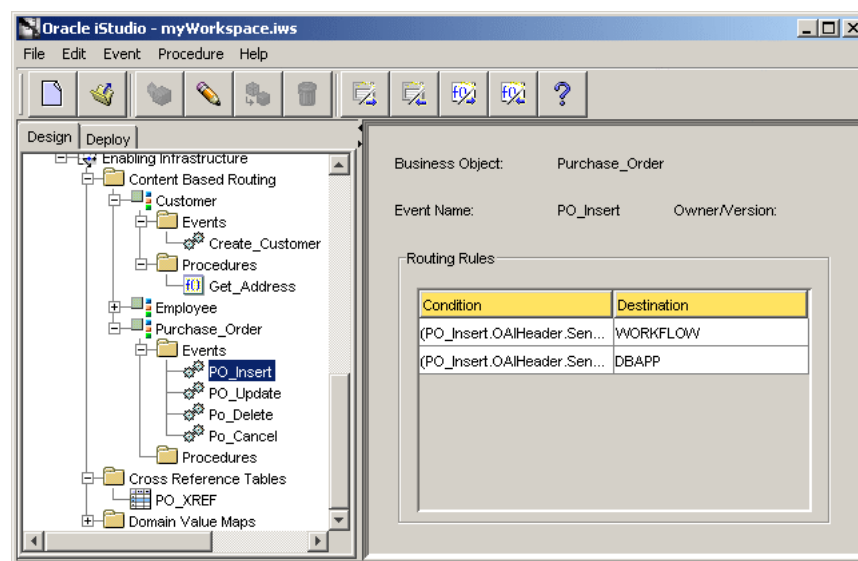
When an event is published, it is automatically routed to any event's subscriber, by default. If the routing of an event needs to be based on a value in the message or message header, then content-based routing is required in this scenario. All changes to the purchase orders must be approved and routed to Oracle Workflow to apply business logic.

The logic to be applied for the Events PO_Insert, PO_Update, and PO_Delete as follows:

- If AQAPP is the source application, then route to the WORKFLOW destination application. The Wizard steps are as follows:
 1. Source Page: Select OAI_Header.SendingApplication
 2. Chose Operator Page: Select =
 3. Chose Value Page: Enter AQAPP
 4. Addition Condition Page: Select Radio Button Complete & press Finished
 5. Destination Page: Select WORKFLOW
- If WORKFLOW is the source application, then route to the DBAPP destination application. The Wizard steps are as follows:
 1. Source Page: Select OAI_Header.SendingApplication
 2. Choose Operator Page: Select =
 3. Chose Value Page: Enter WORKFLOW
 4. Addition Condition Page: Select Radio Button Complete & press Finished
 5. Destination Page: Select DBAPP

The procedure repeats for the PO_Update and PO_Delete events. [Figure A-12](#) describes the completed content-based routing in iStudio.

Figure A-12 Completed Content Routing in iStudio



A.3.10 Create an Oracle Workflow Process Bundle

A process bundle enables related business processes to be grouped and transferred to the Oracle Workflow environment where user-defined business logic is applied.

Each business process enables related publish, subscribe, invoke, and implement activities to be grouped and placed in the Oracle Workflow Business Event System.

Create a Process Bundle

The following steps describe creating the PO process bundle using iStudio:

1. From the project list, expand the Workflow node and navigate to Process Bundle.
2. Right-click **Business Processes** and select **New**. The Create Process Bundle dialog is displayed.
3. Enter **PO** in the Process Bundle Name field and click **OK**.

Create a Business Process

The following steps describe creating the PO business process using iStudio:

1. Expand the Process Bundle node on the project list and navigate to Business Processes.
2. Right-click **Business Processes** and select **New**. The Create Business Process dialog is displayed.
3. Enter **PO** in the Business Process Name field and click **OK**.

Create the Subscribe and Publish Activities

The Oracle Workflow business process uses the common view. As a result, transformation and mapping is not required and the only types of activities used are as follows:

- **Subscribe:** Oracle Workflow receives a message from OracleAS Integration InterConnect.
- **Publish:** Oracle Workflow sends a message to OracleAS Integration InterConnect.
- **Invoke:** Oracle Workflow sends a request message to OracleAS Integration InterConnect and receives a reply.
- **Implement:** Oracle Workflow receives a request from OracleAS Integration InterConnect and sends a reply.

In this scenario, the **PO_Insert**, **PO_Update**, and **PO_Delete** messages are routed to Oracle Workflow to apply business logic. Based on this logic, messages are sent to the Order Fulfillment Application or the **PO_Cancel** message is sent to the Legacy Application. Oracle Workflow must:

- Subscribe to and publish **PO_Insert**.
- Subscribe to and publish **PO_Update**.
- Subscribe to and publish **PO_Delete**.
- Publish **PO_Cancel**.

Create Subscribe Activity

The following steps describe creating the subscribe activity using iStudio:

1. From the Project list, expand the Workflow node and navigate to Business Processes.

2. Right-click PO business process and select **Subscribe Activity**. Right-click any item to display a dialog.
3. Select Event PO_Insert and click **OK**.

Repeat these steps for the PO_Update and PO_Delete events, substituting the correct values where necessary.

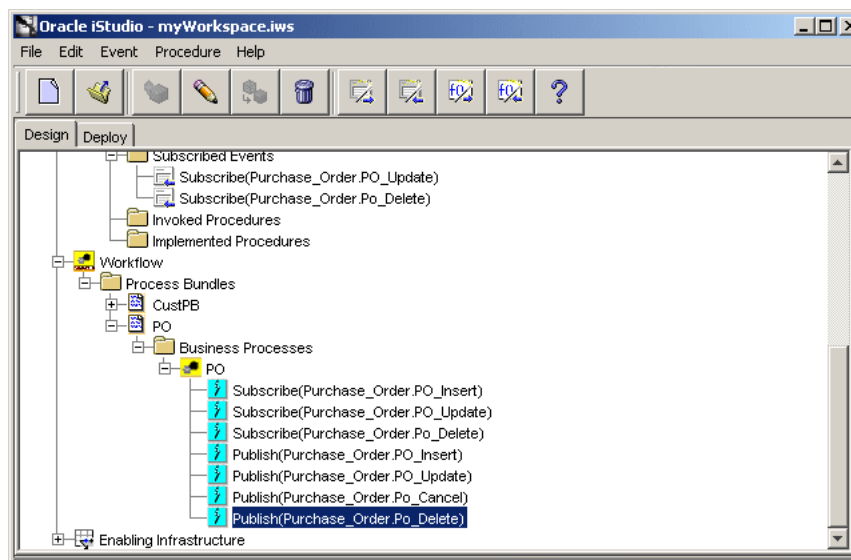
Create Publish Activity

The following steps describe creating the publish activity using iStudio:

1. From the Project list, expand the Workflow node and navigate to Business Processes.
2. Right-click PO business process and select **Publish Activity**. Right-clicking any item displays a pop-up box.
3. Select Event PO_Insert and click **OK**.

Repeat these steps for the PO_Update, PO_Delete, and PO_Cancel events, substituting the correct values where necessary. The subscribe and publish events appear in the Design Object Navigator under the PO node as shown in [Figure A-13](#).

Figure A-13 *Subscribe and Publish Activities in iStudio*



A.3.11 Deploy the Process Bundle to Oracle Workflow

Deploying the Oracle Workflow process bundle accomplishes the following:

- Places the event definitions in the Oracle Workflow Business Event System.
- Creates a default Oracle Workflow file (.wft).
- Launches the Oracle Workflow Builder and Monitor.

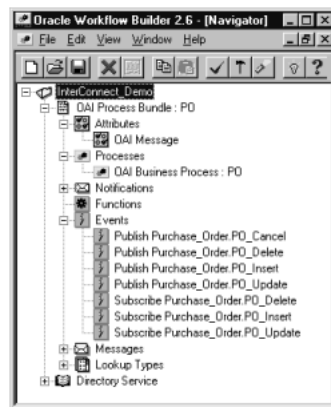
See Also: [Chapter 7, "Using Oracle Workflow"](#)

The following steps describe deploying the process bundle to Oracle Workflow:

1. Right-click the Workflow node on the Deploy tab in iStudio and select **Deploy**. The Deploy dialog is displayed.

2. Select Event Definitions to Workflow Business Event System, then Process Definitions for File in the Deploy to Workflow box.
3. Click **OK**. The Workflow BES Login dialog is displayed.
4. Log in to Oracle Workflow using the correct username, password, and URL. Click **OK**. The Deploy dialog is displayed.
5. Enter a file name for the Oracle Workflow file, such as InterConnect_Demo.wft, in the File Name field, and click **Open**. Oracle Workflow is started with InterConnect_Demo as shown in [Figure A-14](#).

Figure A-14 Completed Deployment in Oracle Workflow



A.3.12 Creating Objects in Oracle Workflow for Modeling

The original requirement for this scenario are as follows:

"An administrator must approve all changes such as insert, update, and delete before they are applied to the Order Fulfillment System. If a change is approved, it is sent to the Order Fulfillment System. If a change is rejected, then a cancellation notification is sent back the legacy system."

This business logic can be implemented in Oracle Workflow. The Oracle Workflow components required are:

- An Item Type equivalent to a Project
- An Attribute An object to hold the message in the event
- A Process To model the Business Logic
- Events For the modeling in the process.
- A Notification To notify the administrator in the Oracle Workflow Monitor.

Components transferred from iStudio.

- Item Type: OAI Process Bundle: PO

- Attribute: OAI Message
- Process: OAI Business Process: PO
- Events:
 - Publish Purchase_Order.PO_Cancel
 - Publish Purchase_Order.PO_Insert
 - Publish Purchase_Order.PO_Update
 - Publish Purchase_Order.PO_Delete
 - Subscribe Purchase_Order.PO_Insert
 - Subscribe Purchase_Order.PO_Update
 - Subscribe Purchase_Order.PO_Delete

Oracle Workflow components are required to create a Notification.

A.3.12.1 Message

The message a notification activity will send.

A.3.12.2 Lookup Type

A static list of values that can be referenced by various objects. For example a message attribute can reference a lookup type as a means of providing a list of possible responses to the performer of a notification.

A.3.12.3 Notification

When the workflow engine reaches a notification activity, it issues a Send() API call to the Notification System to send the message to an assigned performer. When a performer responds to a notification activity, the Notification System processes the response and informs the workflow engine that the notification activity is complete.

A.3.12.4 What Oracle Workflow provides.

Oracle Workflow has a set of pre-defined item types with standard functionality. The Standard item type contains generic activities that can be copied in a user's item type. In this scenario we will be using the Lookup Type Approval.

A.3.12.5 Copy Lookup Type (Approval)

As described, the user must create a Oracle Workflow Notification. The notification has two dependent objects, A lookup Type and a Message. The Lookup Type (Approval) can be copied from the standard item type.

A.3.12.6 Create an Oracle Workflow Message

The following steps describe creating a new Oracle Workflow message called Insert_Message

1. In the Object Navigator right-click the Message Node and select New to launch the property sheet. In each tab, add the following entries:
2. Message Tab:
 - Internal Name: Insert_Message
 - Display Name: Insert Message

- Description: Insert Message
3. Body Tab:
 - Subject: Insert Message
 - Text Body: A record has been Inserted in the Purchase Order Table.
 4. Result Tab:
 - Display Name: Insert_Message
 - Description: Insert_Message
 - Lookup Type: Approval (From Lookup Type)
 5. Click **OK**.

Using the default Copy and Paste functionality create the following messages using message Insert_Message as the template:

- Update_Message: Repeats the preceding steps and use the same setting, changing all references to insert to update.
- Delete_Message: Repeats the preceding steps and use the same setting, changing all references to insert to Delete.

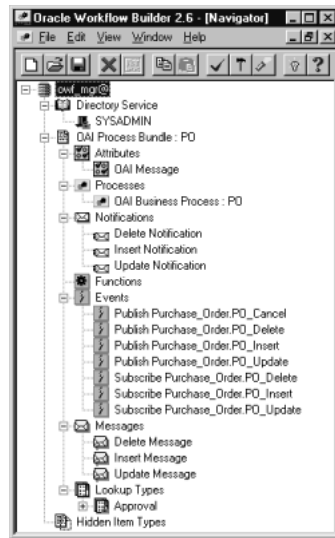
A.3.12.7 Create an Oracle Workflow Notification

The following steps describe creating a new Oracle Workflow Notification.

1. In the Object Navigator, right-click the Notification Node and select **New** to launch the property sheet. In each tab, add the following entries:
2. Activity Tab:
 - Internal Name: Insert_Notification
 - Display Name: Insert_Notification
 - Description: Insert_Notification
 - Message: Insert_Message (Created previous step)
 - Result Type: Approval (From Lookup Type)
3. Click **OK**.

Using the default Copy and Paste functionality create the following notifications using notification Insert_Notification as the template:

- Update_Notification: Repeats the preceding steps and use the same setting, changing all references to insert to update.
- Delete_Notification: Repeats the preceding steps and use the same setting, changing all references to insert to delete.

Figure A-15 Completed Oracle Workflow Notifications

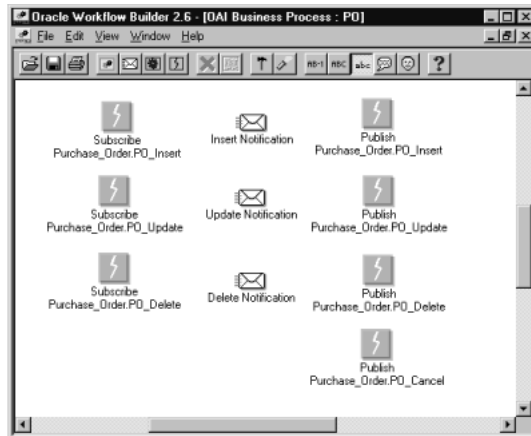
A.4 Modeling Business Logic in Oracle Workflow

Now that all of the required objects have been created, the business logic can be modeled. The following steps describe this process.

1. In the Oracle Workflow Object Navigator, expand the OAI Process Bundle: PO item type.
2. Expand the Processes node.
3. Right-click OAI Business Process: PO and select **Process Details**.

Another way to display the process details is to double-click OAI Business Process: PO.

4. Move the following from the Oracle Workflow Object Navigator to the Oracle Workflow Workspace:
 - Insert_Notification
 - Update_Notification
 - Delete_Notification
5. Rearrange the items as shown in [Figure A-16](#).

Figure A-16 Items Arranged in Oracle Workflow Builder

1. The subscribe events are the entry point in this process. The Start and End Property for each event must be edited and set to START. Double-click an object to launch its property sheet. The Start and End property is under the Node tab.

Subscribe to the following events:

- Purchase_Order.PO_Insert
- Purchase_Order.PO_Update
- Purchase_Order.PO_Delete

2. The publish events are the exit point from this process. The Start and End Property for each event must be edited and set to END. Double-click an object to launch its property sheet. The Start and End property is under the Node tab.

Publish to the following events:

- Purchase_Order.PO_Insert
- Purchase_Order.PO_Update
- Purchase_Order.PO_Delete
- Purchase_Order.PO_Cancel

3. The notifications must be assigned in order for a person to receive the notification in Oracle Workflow Monitor. The Performer value property should be set to SYSADMIN for each notification. Double-click an object to launch its property sheet. The Performer Value field is located under the Node tab.

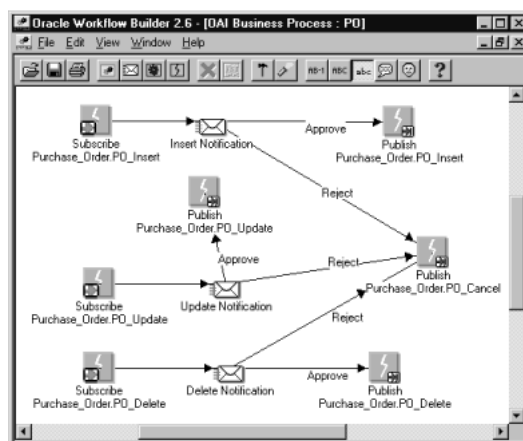
Assign the following notifications:

- Insert_Notification
- Update_Notification
- Delete_Notification

4. Mapping lines need to be drawn between the objects to define the process flow. Lines are drawn by right-clicking on an object and dragging to another object.

- a. Draw a mapping line from Subscribe Purchase_Order.PO_Insert to Insert_Notification.
 - b. Draw a mapping line from Insert_Notification to Publish Purchase_Order.PO_Insert and select **Approve** from the list that will appear when the line is drawn.
 - c. Draw a mapping line from Insert_Notification to Publish Purchase_Order.PO_Cancel and select **Reject** from the list that will appear when the line is drawn.
 - d. Repeat steps for Update & Delete objects.
5. Save your work to the database. The completed business process in Oracle Workflow Builder is shown in [Figure A-17](#).

Figure A-17 Completed Business Process in Oracle Workflow Builder



A.5 Deployment

After modeling the business objects, it is time to deploy the OracleAS Integration InterConnect business objects. The following section describes the deployment process.

- [Setting Queues](#)
- [Sync Adapters](#)
- [Exporting and Installing Code](#)

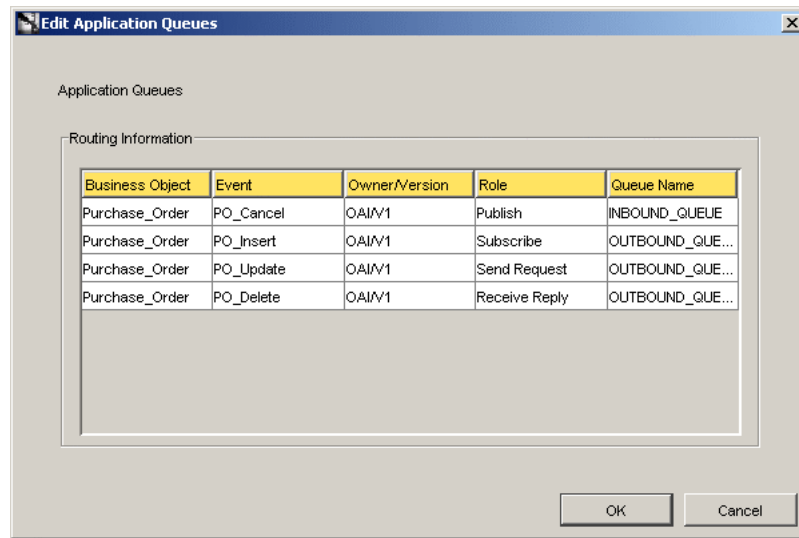
A.5.1 Setting Queues

The AQAPP application in iStudio corresponds to the Advanced Queuing adapter that communicates with the legacy application. The legacy application, through a database trigger, places inserted, updated, and deleted records into a queue using Oracle Advanced Queuing. To communicate to and from the OracleAS Integration InterConnect environment, the adapter must be configured to send and receive on those external queues.

The following steps describe this task.

1. On the Deploy tab in iStudio, expand the Applications list and navigate to AQAPP.
2. Expand the AQAPP node and navigate to the Routing node.
3. Expand the Routing node and select **Application Queues**. The Application Queues property sheet is displayed in the iStudio window.
4. Select **Edit** from the Edit menu. This will launch the Edit Application Queue dialog as shown in [Figure A-18](#).

Figure A-18 Application Queues in iStudio



5. Add the Queue name to each event:

Queue Name	Event
INBOUND_QUEUE	PO_Cancel
OUTBOUND_QUEUE	PO_Insert, PO_Update, and PO_Delete

6. Click **OK**.

A.5.2 Sync Adapters

Each adapter has different cache settings to minimize communication to the repository and to improve performance. If you have updated the metadata, you must synchronize the adapter and repository metadata. The following steps describe this task:

1. Select **File** from the menu, then select **Sync Adapters**. The Sync Adapters dialog is displayed.
2. Select the applications to which to sync adapters, and click **OK**.

A.5.3 Exporting and Installing Code

Depending on the adapter type, there may be code that must be exported to a file and installed in the target application database. The following steps describe exporting the code using the Export Application dialog in iStudio.

1. In the iStudio window, click the **Deploy** tab. Right-click **Applications** and select **Export PL/SQL**. The Export Application dialog is displayed.
2. Select the applications to export code.
3. Enter the file prefix in the File Prefix field and click **OK**.

The resulting text file is a SQL*Plus script that is run on the target schema.

See Also: ["Deploying PL/SQL Stored Procedures"](#) on page 8-1

Example A-1 Exporting and Installing Code

The following example helps to explain exporting and installing code. This example is based on the following:

- Adapter type: Database Adapter
- iStudio application: DBAPP
- Subscribe event: PO_Delete

```
PROCEDURE sub_PO_Delete_OAI_V1 (   POID           IN NUMBER,
                                POITEM          IN LONG,
                                PRICE           IN LONG,
                                QUANTITY       IN NUMBER,
                                LAST_UPDATED    IN DATE, )
AS
    v_poid NUMBER :=poid;
BEGIN
    DELETE FROM PO WHERE v_poid = poid;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN NULL;

END sub_PO_Delete_OAI_V1;
```

A.6 Conclusion

The final step is to test the integration.

1. A record is inserted into the Legacy System.
2. The legacy system's database trigger queues the record in its OUTBOUND_QUEUE.
3. OracleAS Integration InterConnect receives the message, performs transformations, converts data to a common view, and routes the message to Oracle Workflow.
4. Oracle Workflow applies the business logic and issues a notification.
5. The System Administrator logs on to Oracle Workflow Monitor, receives the Insert_notification, and approves the record.
6. OracleAS Integration InterConnect received the message, performs transformations, cross-references the primary keys, converts data to the application View, and routes the message to Order Fulfillment System.

7. The deployed code receives the message and inserts the record into the Order Fulfillment system.
8. Oracle InterConnect Manager is used to examine the inserted record and monitor the integration throughput.

This process should be repeated for Update and Delete.

Using the Data Definition Description Language

This appendix describes how to use the Data Definition Description Language (D3L) in its native format message to application view translations and vice-versa. It contains the following topics:

- [About D3L](#)
- [Native Format Message and D3L File Example](#)
- [D3L File Structure](#)
- [D3L Integration with OracleAS Integration InterConnect Adapters](#)
- [Installing D3L](#)
- [Configuring D3L](#)
- [D3L Use Case](#)
- [Additional D3L Sample Files and DTD](#)

B.1 About D3L

This section contains these topics:

- [What Is D3L?](#)
- [When Is D3L Used?](#)

B.1.1 What Is D3L?

D3L is an XML-based message description language. It describes the structure that an application's native, non-XML format message (also called the native view of the application) must follow to communicate with OracleAS Integration InterConnect. Oracle provides the following OracleAS Integration InterConnect transport adapters that interact with the D3L message description language:

- FTP
- HTTP
- MQ Series
- SMTP

OracleAS Integration InterConnect Adapters perform the following tasks:

- Validate the D3L message description files during runtime initialization.

- Use the D3L translation engine (subcomponent of the bridge) to translate messages from:
 - Native format message to application view
 - Application view to native format message
- Transport message payload data between an application and OracleAS Integration InterConnect.

Note: Native format messages that are already in XML format are not translated by OracleAS Integration InterConnect Adapters if the `ota.type` parameter is set to `XML` in the `adapter.ini` file.

See Also: ["D3L Integration with OracleAS Integration InterConnect Adapters"](#) on page B-23

B.1.2 When Is D3L Used?

Some applications do not use XML as their native message payload format. These applications use native formats, such as structured records of bytes and characters. For these native formats to be successfully translated into a format understood by other applications, the content of their messages must follow a predefined, structured set of rules. This structured format can then be translated into an application view, transformed into a common view, and understood by other applications.

D3L provides both a predefined, structured set of rules and translation capabilities for native format messages. D3L provides:

- An XML-based message description language that describes the contents of native format messages
- A translation engine that uses the instructions defined in the D3L file to translate the native format message contents to and from an application view

The D3L descriptions must comply with a syntax defined by the D3L document type definition (DTD). D3L enables you to describe the record layout of binary, string, structured, and sequence data. Use D3L only when the number of fields in the underlying native format message is fixed and known. D3L is not suitable for the following:

- Descriptions of arbitrarily structured data like regular XML
- Name-value pair data
- Conditional data structures, which require token look-aheads to parse

See Also:

- ["Supported D3L Data Types"](#) on page B-10
- ["D3L DTD"](#) on page B-52

B.1.3 D3L Features

This section describes data definition description language enhancements. It includes the following topics:

- [Integrate Transport Properties](#)
- [Allow Multiple Imparrays](#)

B.1.3.1 Integrate Transport Properties

This enhancement allows the Data Definition Description Language (D3L) author to add a new type of member, property. Syntax, to a D3L structure, in addition to fields and pads. For example:

```
<struct ...
  ...
  <property name="prop_name" />
  ...
</struct>
```

Note: There is no type definition associated with this structure element.

The modified D3L Data Type Definition (DTD) for this new structure element is:

```
<!ENTITY % StructElements
  "field | property | pad"
>

<!ELEMENT property EMPTY >
  <!ATTLIST property
    %FieldAttributes;
>
```

The semantics of this new structure element is to link data in a transport protocol header with the message payload. In other words, when a D3L containing a structure with one, or more, property member(s) is imported in iStudio, it will create a String OAI attribute with the name specified in the property name attribute.

At runtime, this OAI attribute will be populated with the value of a transport protocol header, `inbound-to-hub`, which name matches the name attribute of the property member. Similarly, for outbound messages, the OracleAS Integration InterConnect message payload property value will define the value of the corresponding protocol header.

For example, if using OracleAS Integration InterConnect Adapter for FTP, the file structure would be:

```
<struct ..>
  <property name="filename" />
</field ...
```

In this case, the OracleAS Integration InterConnect Application View attribute `filename`, that is derived from the D3L definition, would be assigned the name of the actual file being passed to D3L. For outbound message, the value will determine the physical filename being used to store the file.

If using OracleAS Integration InterConnect Adapter for HTTP, an example file structure would be:

```
<struct ..>
  <property name="Host" />
  <property name="Referer" />
</field name="..." < ...
```

Note: This is a dynamic payload dependent feature, which will override settings in the `adapter.ini` file and/or Application View Meta Data Modify Fields. As a result, the property `ota.send.endpoint` could be overridden by a corresponding message attribute defined through the D3L.

B.1.3.2 Allow Multiple Imparrays

The D3L syntax allows you to create multiple nested imparrays for outbound translations (`app-to-native` or `hub-to-spoke`).

Intuitively it makes sense to allow multiple nested imparrays to match multiple nested arrays in XML as XML does not have the need to declare the length of an array. For example:

```
<array1>
  <array2>..</array2>
  <array2>..</array2>
</array1>
<array1>
  <array2>..</array2>
  <array2>..</array2>
<array2>..</array2>
```

If this XML message was published by OracleAS Integration InterConnect Adapter for AQ, and consumed by OracleAS Integration InterConnect Adapter for FTP that is running in D3L mode, the preceding structure would then be matched by the following D3L structure:

```
<imparray id="array1">
  <imparray id="array2">
    ..
```

As D3L does not perform parsing for `app-to-native` translation, also known as `production`, the preceding D3L is entirely possible. However, for `native-to-app` translations, the preceding D3L would be invalid as a single imparray by itself would consume the rest of the native message.

Note: This new imparray semantics depart from the design principle that D3L is a fully bidirectional symmetric translator and can perform both `native-to-app` and `app-to-native` translation using just one the same D3L definition.

Finally, the D3L translator will determine, at runtime, whether multiple nested imparrays exist in a D3L. If multiple nested imparrays are detected, the translator will prevent the D3L from being used for parsing purposes (`native-to-app` translations). If not detected, the D3L translator will flag an error condition.

B.2 Native Format Message and D3L File Example

This section provides an example of how the contents of a native format message are:

- Described in a D3L file
- Configured with the required D3L file

To successfully translate the native format message, you need to satisfy both the preceding conditions.

This section contains the following topics:

- [Description of Native Format Message Contents in a D3L File](#)
- [Configuration of Native Format Message with a D3L File](#)

B.2.1 Description of Native Format Message Contents in a D3L File

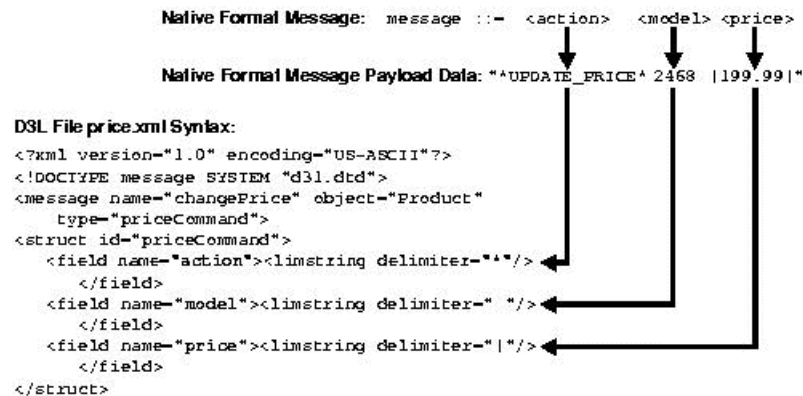
This section describes an application's native format message (named `price`) that contains data for updating the price of personal computer model number 2468 to 199.99. The native message uses the following format to describe the data:

```
message ::= action model price
```

Where...	Is...
action	UPDATE_PRICE
model	2468
price	199.99

The data must strictly follow the structure defined in a D3L file, for this example, `price.xml`. This D3L translation engine translates the data into an application view. [Figure B-1](#) shows how a D3L file defines the structure that the native format message `price` must follow to successfully define the data elements.

Figure B-1 Native Format Message Payload Data and D3L File Syntax



All three data elements are defined as strings with different delimiters for separating their data.

B.2.2 Configuration of Native Format Message with a D3L File

When the D3L translation engine receives a native format message, such as `price`, it must determine the exact D3L file to verify the native format message contents, such as, `price.xml`.

This section describes the methods for configuring the correct D3L file with the native format message. It contains the following topics:

- [adapter.ini Parameter File Setting](#)
- [Message Header Attributes](#)

B.2.2.1 adapter.ini Parameter File Setting

The `ota.d3ls` parameter in the `%ORACLE_HOME%\integration\interconnect\adapters\application\adapter.ini` file enables you to define the D3L file to use with the native format message. For example:

```
ota.d3ls=price.xml
```

When the D3L translation engine receives the native format message from the bridge, it retrieves the correct D3L file based on this parameter setting. Multiple D3L files can also be defined using the `ota.d3ls` parameter. For example:

```
ota.d3ls=price.xml,emp.xml,booking.xml
```

Unless one of the methods described in "[Message Header Attributes](#)" is used, the D3L translation engine compares the data structure in the native format message to each D3L file until it finds the correct one to use for translation.

B.2.2.2 Message Header Attributes

The D3L file includes message header attributes work with the D3L engine to choose the correct D3L file for translating a native format message to an application view. The values for these message header attributes match the settings in the native format message.

Message header attribute values override the approach of comparing each D3L file defined with the `ota.d3ls` parameter in the `adapter.ini` file with a native format message.

Two methods to set the message header attribute values are available:

- [Name/Value Pair Message Header Attributes](#)
- [Magic Value Message Header Attribute](#)

Both methods enable the D3L translation engine to use the correct D3L file for translation after receiving the native format message.

Note: When the correct D3L file is selected and a successful translation has taken place, the `message` element attributes `name` and `object` in the D3L file define the OracleAS Integration InterConnect event name and business object, respectively.

B.2.2.2.1 Name/Value Pair Message Header Attributes

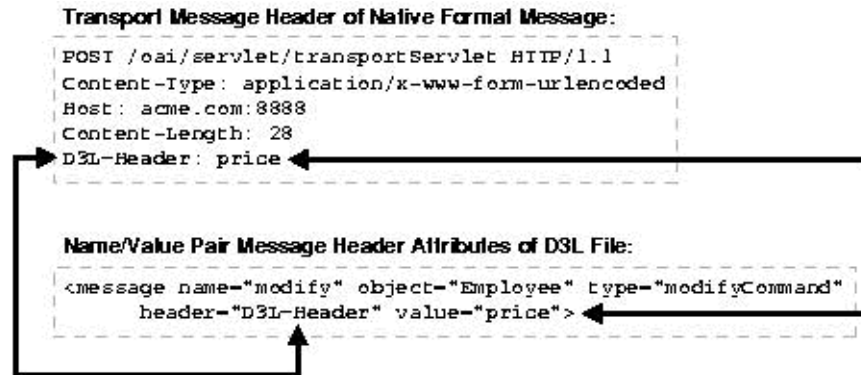
OracleAS Integration InterConnect Adapters, such as the HTTP adapter, make their protocol level transport properties available to the D3L translation engine, including custom properties added by a sender application, such as an HTTP client. The D3L file message element enables the user to specify two attributes, `header` and `value`, which match the protocol level headers in a received native format message.

For example, a third-party application uses the custom transport header D3L-Header to communicate with the D3L translation engine which D3L file to use to translate an incoming native format message. The following steps must be performed to set custom values in the transport header:

- Set the D3L-Header parameter in the transport message header to a value that matches the value attribute setting of the <message> element in the D3L file.
- Set the header attribute of the <message> element in the D3L file to D3L-Header to match the D3L-Header parameter name in the transport message header.

Figure B-2 illustrates using the HTTP adapter where D3L-Header and price are the header name and header value, respectively. The header name and value are used to match a native format message with the correct D3L file. The D3L translation engine retrieves the correct D3L file based on these settings.

Figure B-2 Name/Value Pair Message Header Attributes



The D3L engine supports a rudimentary pattern matching capability in the value attribute of the D3L message element.

A D3L author can create a D3L definition, such as:

```
<message type="CrtCust" header="filename" value="cust_create%" ...
```

This definition is for the FTP adapter, which provides a header property called filename that holds the name of a received file.

The preceding D3L destination will be selected to parse incoming files whose filenames match the name pattern in the value attribute, such as,

```
cust_create01
cust_create02
po_int_ext01
cust_create03
po_int_ext02
```

The "wildcard" character in the pattern ("%") can only appear at two places in the attribute string-value, as either the first or the last character, or both the first and last characters.

For example, the following patterns are acceptable:

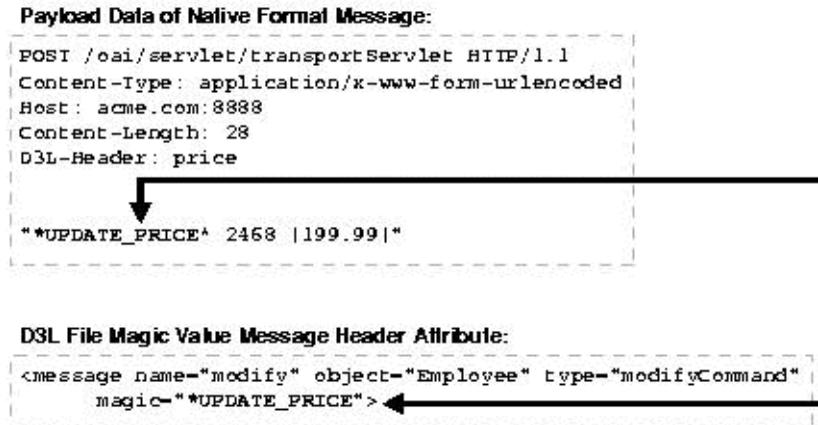
- %endstring: such as %.csv
- startstring%: such as po_int%
- %substring%: such as %create%

B.2.2.2.2 Magic Value Message Header Attribute You can set the magic attribute of the message element in the D3L file to match the first *n* bytes of data in a native format

message. This feature enables you to define the D3L file to use with the native format message. When a native format message is received by the D3L translation engine, the magic values of all D3L files are compared with the first *n* bytes of the native format message. The magic values must be long enough to be unique across all registered D3Ls for a given adapter instance.

Figure B-3 provides an example where *UPDATE_PRICE is the value that configures the native format message with the correct D3L file.

Figure B-3 Magic Value Message Header Attribute



The D3L translation engine retrieves the correct D3L file based on these settings.

The D3L attribute `startsat` of the message element enables the D3L author to specify the byte location to start the magic matching.

Having this attribute allows the D3L definition:

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE message SYSTEM "d3l.dtd">

<message name="newBook" type="BookType" object="BookObj" magic="ISBN#"
  startsat="12">
```

This D3L definition will trigger if a native message contains the byte character sequence ISBN# in byte positions 12 to 16.

See Also:

- ["Native Format Message to Common View Incoming Message Translations"](#) on page B-24
- ["Additional D3L Sample Files and DTD"](#) on page B-48

B.3 D3L File Structure

This section describes the contents of a sample D3L file named `book_reply.xml`.

```
1 <?xml version="1.0" encoding="US-ASCII"?>
2 <!DOCTYPE message SYSTEM "d3l.dtd">
3 <message name="replyFlight" type="BookingReplyType" object="Booking"
4 header="D3L-Header" value="replyOptions">
5   <unsigned4 id="u4" />
6   <unsigned2 id="u2" />
```

```

7     <struct id="DateTimeRecord">
8         <field name="DateInfo">
9             <date format="MMDDYY">
10                <pfxstring id="datstr" length="u4" />
11            </date>
12        </field>
13        <field name="TimeHour"><limstring delimiter="*" /></field>
14        <field name="TimeMinute"><limstring delimiter="*" /></field>
15    </struct>
16    <struct id="ItinRecord">
17        <field name="DepartureTime"><typeref type="DateTimeRecord" /></field>
18        <field name="ArrivalTime"><typeref type="DateTimeRecord" /></field>
19    </struct>
20    <pfxarray id="ItinArray" length="u2">
21        <typeref type="ItinRecord" />
22    </pfxarray>
23    <struct id="BookingReplyType">
24        <field name="AirportCodeFrom"><limstring delimiter="*" /></field>
25        <field name="AirportCodeTo"><limstring delimiter="*" /></field>
26        <field name="Itineraries"><typeref type="ItinArray" /></field>
27    </struct>
28 </message>

```

Lines 1 and 2

These lines define standard information, such as the Prolog and Document Type Declaration (DTD) that must have the specified values (for example, specifying `d31.dtd` as the DTD).

Lines 3 and 4

These lines define the following message element attributes:

- **name:** It must correspond to the associated Oracle Application Server Integration InterConnect application view event name defined in iStudio. The D3L file can also be imported in iStudio when defining the message attributes of an event (the name of which must match the name attribute of the D3L message element).
- **type:** It names a structure that is defined in subsequent lines of this D3L file.
- **object:** It must match the Oracle Application Server Integration InterConnect business object defined in iStudio.
- **header:** It is identified in the set of protocol level transport message headers associated with a native format message.
- **value:** It must match the actual value of the corresponding protocol level transport message header defined through the `header` attribute.

See Also:

- ["Creating Business Objects"](#) on page 3-2
- ["Creating Events"](#) on page 4-2
- ["Name/Value Pair Message Header Attributes"](#) on page B-6
- ["Task 6: Import a D3L File in iStudio"](#) on page B-29

Lines 5 and 6

These lines define an unsigned, four-byte integer and unsigned, two-byte integer. These data type declarations are named `u4` and `u2`, respectively, so they can be referred to later.

Lines 7 Through 15

These lines define the following fields of a structure named `DateTimeRecord`:

- `DateInfo` defines a date format of `MMDDYY` and a length prefixed by an unsigned four-byte integer.
- `TimeHour` defines a string delimited by the character `*`.
- `TimeMinute` defines a string delimited by the character `*`.

Lines 16 Through 19

These lines define the fields of the structure named `ItinRecord`. The fields `DepartureTime` and `ArrivalTime` both consist of the `DateTimeRecord` structure.

Lines 20 Through 22

These lines define a length-prefixed array named `InitArray`, where each array element is of type `ItinRecord`.

Lines 23 Through 28

These lines define the following fields of the message structure `BookingReplyType`, which satisfies the `BookingReplyType` type declaration in the message document element:

- `AirportCodeFrom` is a string delimited by the character `*`.
- `AirportCodeTo` is a string delimited by the character `*`.
- `Itineraries` is a field of type `ItinArray`, which is an array of `ItinRecord`.

B.3.1 Supported D3L Data Types

D3L supports use of the following data types and declarations in a D3L file:

- [Signed or Unsigned Integers](#)
- [Floating Point Numbers](#)
- [Strings](#)
- [Structures](#)
- [Sequences](#)

B.3.1.1 Signed or Unsigned Integers

D3L supports signed or unsigned integers that can be one, two, four, or eight octets in size, and in big or little endian octet ordering.

Example B-1 Quantity Field

```
<field name="quantity">
  <unsigned4 endian="big" align="6"/>
</field>
```

The `quantity` field defines a four byte unsigned binary integer, using big (default) endian, and at an alignment of 6 bytes. For example, D3L will ensure that the reading or writing of this integer will start at a position in the buffer, so that $\langle position \rangle \text{ modulus } \langle alignment \rangle = 0$.

Note: *Little Endian* means that the low-order byte of the number is stored in memory at the lowest address, and the high-order byte at the highest address. whereas *Big Endian* means that the high-order byte of the number is stored in memory at the lowest address, and the low-order byte at the highest address.

Data example

Byte addresses (hex):

00 01 02 03 04 05 06 07 08 09 0A 0B

Byte (hex):

00 00 00 00 00 00 **80 FF FF FF** 00 00

Parsed value (dec):

$quantity = 128 \times 256^3 + 255 \times 256^2 + 255 \times 256^1 + 255 \times 256^0 = 2164260863$

Example B-2 Weight and Length Field

```
<field name="weight"> <unsigned2 align="3"/> </field>
<field name="length"> <unsigned2 align="3"/> </field>
```

The weight and length fields define two 2-byte unsigned binary integers, using big endian, and an alignment of 3 bytes.

Data example

Byte addresses (hex):

.. 07 08 09 0A 0B 0C 0D 0E ..

Byte (hex):

.. 00 00 **EE 88** 00 **22 F0** 00 ..

Parsed value (dec):

$weight = 238 \times 256^1 + 136 \times 256^0 = 61064$

$length = 34 \times 256^1 + 240 \times 256^0 = 8944$

Example B-3 Temperature and Pressure Field

```
<field name="temperature"> <signed2 endian="little" /> </field>
<field name="pressure"> <unsigned4 endian="big" /> </field>
<field name="wind"> <unsigned2 endian="little" align="4" /> </field>
```

The temperature field defines a 2-byte signed binary integer, using little endian, and no alignment.

The `pressure` field defines a 4-byte unsigned binary integer, using big endian, and no alignment.

The `wind` field defines a 2-byte unsigned binary integer, using little endian, and a 4-byte alignment.

Data example

Byte addresses (hex):

```
.. 60 61 62 63 64 65 66 67 68 69 ..
```

Byte (hex):

```
.. EF FE 00 00 04 0A 00 00 3C 00 ..
   ^         ^         ^
   little end big end alignment
```

Parsed value (dec):

```
temperature = 256x256 - (239x2560 + 254x2561) = -273
pressure    = 4x2561 + 10x2560 = 1034
wind        = 60x2560 + 0x2561 = 60
```

B.3.1.2 Floating Point Numbers

D3L supports single- and double-precision, IEEE format, floating-point data.

Single-precision floating point numbers (floats) take up four bytes or octets.

Double-precision floating point numbers (doubles) take up eight bytes or octets.

Example B-4 Distance and Age Field

```
<field name="distance"> <double align="6"/> </field>
<field name="age"> <float /> </field>
```

The `distance` field defines an 8-byte double-float (floating-point value according to the IEEE 754 floating-point double precision bit layout), at an alignment of 6 bytes.

The `age` field defines a 4-byte single-float (floating-point value according to the IEEE 754 floating-point single precision bit layout).

Note: The IEEE 754 floating-point format is parsed and produced by the following Java class methods:

```
java.io.DataInput.readFloat()
java.io.DataInput.readDouble()
java.io.DataOutput.writeFloat()
java.io.DataOutput.writeDouble()
```

Data example

Byte addresses (hex):

```
.. 77 78 03 04 05 06 07 08 09 0A 0B xx xx ..
```

Byte (hex):

```
00 00 D2 47 D3 CE 16 2A B1 A1 5E 5D 6B 0B ..
      ^ double                ^ float
```


Parsed value (dec):

```
distance = 1 x 1038
age = 1 x 1018
```

B.3.1.3 Strings

D3L supports the following string types:

- Constant length strings without delimiters, with optional padding to fill out empty spaces.
- Delimited strings can be delimited by an arbitrary delimiter character.
- Length-prefixed strings where the length prefix is a numeric type. Numeric types are binary integer types described in "Signed or Unsigned Integers" or are a number stored as a string.
- Strings terminated by a specified character.
- Strings terminated by a delimiter defined by an enclosing `limarry` structure.
- Four date formats: `MMDDYY`, `DDMMYY`, `MMDDYYYY`, `DDMMYYYY`, where the information is stored as a string in one of these formats with any separator character between month, date, and year, such as `12!24=01`).
- Numbers not defined as binary data, but as strings. Any string format can define a number (either an integer or a floating-point entity). In iStudio, a D3L field of type number is handled as a double.

Example B-5 Constant length strings

```
<field name="CURRENCY_CODE">
  <padstring length="4" padchar=" " padstyle="tail"/>
</field>
<field name="COUNTRY_CODE">
  <padstring length="2" padchar="" padstyle="none"/>
</field>
<field name="TO_USD_RATE">
  <padstring length="12" padchar="0" padstyle="head"/>
</field>
```

The `CURRENCY_CODE` field defines a fixed length string of four characters. Any blank (" ") characters (pads) near the end (`padstyle="tail"`) of the string are not considered part of the data value.

The `COUNTRY_CODE` field defines a fixed length string of two characters. All characters in this field are part of the data value because `padstyle` is "none".

The `TO_USD_RATE` field defines a fixed length string of 12 characters. Any zeros at the beginning (`padstyle="head"`) of the string are not considered part of the data value.

Data example

Native byte (character) stream:

```
GBP UK000012550.00
```

Parsed values:

```
CURRENCY_CODE = 'GBP'
COUNTRY_CODE  = 'UK'
TO_USD_RATE   = '12550.00'
```

Example B-6 Delimited strings

```
<field name="State"> <limstring delimiter="." /> </field>
<field name="Region"> <limstring delimiter="." /> </field>
<field name="City"> <limstring delimiter="|" /> </field>
<field name="Landmark"> <limstring delimiter="|" /> </field>
<field name="Street"> <limstring delimiter="+" /> </field>
```

The "State", "Region", "City", "Landmark", and "Street" fields are delimited strings enclosed by ".", ".", "|", "|", and "+", respectively.

Data example

Native byte (character) stream:

```
.FL..Florida Keys.|Key West||Ernest Hemingway Museum|+Whitehead St.+
```

Parsed values:

```
State = 'FL'
Region = 'Florida Keys'
City = 'Key West'
Landmark = 'Ernest Hemingway Museum'
Street = 'Whitehead St.'
```

Example B-7 Length prefixed strings

```
<unsigned1 id="ubyte1" />
<unsigned2 id="ubyte2" endian="little" />
<struct>
  <field name="user"> <pfxstring length="ubyte1" /> </field>
  <field name="encr_user"> <pfxstring length="ubyte2" /> </field>
```

The user field defines a string the length of which is defined by a 1-byte binary integer preceding the string contents.

The encr_user field defines a string the length of which is defined by a 2-byte binary integer preceding the string contents.

Data example

Byte addresses (hex):

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12
```

Characters:

```
03 j o e 0D 00 D U Z a c . 1 H K V m I Y
```

Parsed values:

```
user = 'joe'
encr_user = 'DUZac.1HKVmIY'
```

Example B-8 Terminated strings

```
<field name="product"> <termstring endchar="," /> </field>
<field name="ordered"> <termstring endchar="," /> </field>
<field name="inventory"> <termstring endchar="," /> </field>
<field name="backlog"> <termstring endchar="," /> </field>
<field name="listprice"> <termstring endchar="\n"/> </field>
```

Each of the first four fields is populated with input characters until the terminating/ending character (endchar) is encountered (","). The last field ends with a linefeed.

Data example

Native byte (character) stream:

```
1020,16,18,,1580.00<LF>
```

Parsed values:

```
product = '1020'
ordered = '16'
inventory = '18'
backlog = ''
listprice = '1580.00'
```

Note: The backlog field is empty.

Example B-9 Simple strings - simplestring

```
<limarray id="CSV_Type" contchar="," endchar="\n">
  <simplestring/>
</limarray>
<struct>
  <field name="CSV"> <typeref type="CSV_Type"/> </field>
```

The CSV field references a type declaration "CSV_Type" to a delimited array. The array members are separated by comma (contchar=",") and end with linefeed (endchar="\n").

Data example

Native byte (character) stream:

```
5,18,2.5,255,78.75,9
```

Parsed values:

```
CSV[] = { '5', '18', '2.5', '255', '78.75', '9' }
```

Example B-10 Dates - date

```
<field name="StartDate">
  <date format="MMDDYY"> <termstring endchar="\n"/> </date>
</field>
<field name="EndDate">
  <date format="DDMMYY"> <termstring endchar="\n"/> </date>
</field>
<field name="Milestone">
  <date format="MMDDYYYY"> <termstring endchar="\n"/> </date>
</field>
<field name="DueDate">
  <date format="DDMMYYYY"> <termstring endchar="\n"/> </date>
</field>
```

The fields contain dates representing the four date formats.

Data example

Byte stream (characters):

```
11/16/02<LF>
24/11/02<LF>
11/20-2002<LF>
23*11*2002<LF>
```

Parsed values:

```
StartDate = Sat Nov 16 00:00:00 PST 2002
EndDate   = Sun Nov 24 00:00:00 PST 2002
Milestone = Wed Nov 20 00:00:00 PST 2002
DueDate   = Sat Nov 23 00:00:00 PST 2002
```

Note: The D3L parser will accept any character between the DD, MM and YY (YY) characters in the native format, but will always produce the "/" separator when translating from application message format to native message format. Hours, minutes, and seconds are not parseable.

Example B-11 String based numbers

```
<unsigned1 id="u1" />
<pxstring id="HueType" length="u1" />

<struct id="ColorDefinition">
  <field name="Red">
    <number> <padstring length="4" padstyle="head" padchar="0"/> </number>
  </field>
  <field name="Green">
    <number> <pxstring length="u1" /> </number>
  </field>
  <field name="Blue">
    <number> <limstring delimiter="."/> </number>
  </field>
  <field name="Brightness">
    <number> <termstring endchar="|"/> </number>
  </field>
  <field name="Hue">
    <number> <typeref type="HueType"/> </number>
  </field>
</struct>
```

The declared type `u1` is an unsigned 1-byte integer (0-255). The second type declaration `"HueType"` is a length prefixed string, where the string length will be defined in a one-byte binary integer preceding the string contents.

The `Red` field is a number defined as a fixed length string of 4 characters, which can be padded with zeros at the beginning.

The `Green` field is a number defined as a length prefixed string, where the string length will be defined in a one-byte binary integer preceding the string contents.

The `Blue` field is a number defined as a "." delimited string. The string beginning and end is demarcated by ".".

The `Brightness` field is a number defined as a string which is read from the current point until the ending character ("|") is encountered.

The `Hue` field is a number defined as a string of type `"HueType"`.

Data example

Byte addresses (hex) and characters (hex values shown in italics):

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
 0  1  2  8 03  1  2  8  .  2  5  5  .  0  .  7
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E
 5  3  3  3  3  | 08 0  .  6  6  6  6  6  6
```

Parsed values:

```
Red = 128.0
Green = 128.0
Blue = 255.0
Brightness= 0.753333
Hue = 0.666666
```

Note: The parsed numbers always internally become doubles.

B.3.1.4 Structures

D3L supports structured types, such as ordered records containing other data types (predefined or user defined). Types can be nested to arbitrary depth. This means you can use structures of sequences of structures of sequences to any finite depth. Recursive, self referencing, and data structures are not supported in D3L.

All data fields in a message format description must be named. These names are used as Oracle Application Server Integration InterConnect message attribute names. All names within the same structure must be mutually unique.

Within a D3L file the first allowable element is `message`. The `message` element must refer to a `struct` (using the `IDREF` type attribute), which then becomes the top-level data structure of the message.

Example B-12 ColorDefinition Field

```
<message type="ColorDefinition" name="myEV" object="myBO">
  <unsigned1 id="u1" />
  <pxstring id="HueType" length="u1" />
  <struct id="ColorDefinition">
    <field name="Red"> ...
  <field name="Green"> ...
```

Note: The top-level structure can be placed anywhere in the D3L file within the scope of the message element.

B.3.1.5 Sequences

D3L supports sequences, such as arrays of various types. These include:

- Delimited arrays (with arbitrary separator and terminator characters)
- Length-prefixed arrays (where the length is one of the numeric types)
- Fixed-length arrays
- Implicit-length arrays (which use all remaining data in the native format message to the end of the buffer)

The data being sequenced can be any other D3L type (predefined or user defined).

Example B-13 Delimited arrays

```
<field name="members">
  <limarray contchar=";" endchar=".">
    <limstring delimiter="." />
  </limarray>
</field>
```

The `members` field becomes an array of data elements separated by semicolons (`contchar=";"`). The end of the array is marked by a period (`endchar="."`). Each data element in the array is a string delimited by a set of periods (`delimiter="."`).

Data example

Native byte (character) stream:

```
.John.;.Steve.;.Paul.;.Todd..
```

Parsed values:

```
members[] = { 'John', 'Steve', 'Paul', 'Todd' }
```

Example B-14 Length prefixed arrays

```
<unsigned2 id="u2" endian="little" align="4" />
<struct>
  <field name="measurements">
    <pfxarray length="u2" > <signed1 /> </pfxarray>
  </field>
```

The `measurements` field becomes an array of signed 1-byte binary integers (`signed1`). The number of elements in the array is determined by the unsigned two-byte binary integer at the beginning of the array.

Data example

Byte addresses (hex):

```
.. 08 09 0A 0B 0C 0D 0E 0F 10 ..
```

Bytes (hex):

```
.. 06 00 FF A2 6C 24 0E 77 ..
```

Values (dec):

```
measurements[] = { -1, -94, 108, 36, 14, 119 }
```

Example B-15 Fixed length arrays

```
<field name="digits">
  <fixarray length="10">
    <number>
      <termstring endchar="-">
    </number>
  </fixarray>
```

The `digits` field becomes an array of numbers (doubles). Each number element in the native byte format is represented as a string which is terminated by a dash (`endchar="-"`). The number of elements in the array must always be 10 (`length="10"`).

Data example

Native byte (character) stream:

```
1-2-3-4-5-6-7-8-9-0-
```

Parsed values:

```
digits[] = { 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 0.0 }
```

Example B-16 Implicit length arrays

```
<message name="addOrders" object="Order" type="OrdersType">

  <number id="Number"> <termstring endchar="," /> </number>
  <number id="Price"> <termstring endchar=";" /> </number>
  <number id="Total"> <termstring endchar="\n"/> </number>

  <struct id="OrderLineType">
    <field name="LineNo"> <typeref type="Number" /> </field>
    <field name="ProductNo"> <typeref type="Number" /> </field>
    <field name="Quantity"> <typeref type="Number" /> </field>
    <field name="LinePrice"> <typeref type="Price" /> </field>
  </struct>

  <struct id="OrderType">
    <field name="OrderTotal"> <typeref type="Total" /> </field>
    <field name="OrderLines">
      <limarray contchar="\n" endchar="\n\n">
        <struct>
          <field name="OrderLine"> <typeref type="OrderLineType" /> </field>
        </struct>
      </limarray>
    </field>
  </struct>

  <number id="ID"> <termstring endchar="\n" /> </number>

  <imparray id="OrdersArrayType">
    <struct>
      <field name="CustomerID"> <typeref type="ID" /> </field>
      <field name="CustomerName"> <termstring endchar="\n" /> </field>
      <field name="Order"> <typeref type="OrderType" /> </field>
    </struct>
  </imparray>

  <struct id="OrdersType">
    <field name="OrdersArray"> <typeref type="OrdersArrayType" /> </field>
  </struct>
</message>
```

The `OrdersType` structure consists of a single field `OrdersArray` which is an implicit array of three fields: a `CustomerID`, `CustomerName` and `Order` (of type `OrderType`). `OrdersArrayType` is an implicit array. As a result, it will consume all remaining bytes in the native byte input stream. The size of the array is first known when the input byte stream has been exhausted.

The `OrderLines` field is a (nested) array, where each array element is of type `OrderLineType`. The `OrderLineType` is a structure of four fields: `LineNo`, `ProductNo`, `Quantity`, and `LinePrice`.

This input stream must use the following structure to be parseable:

```
CustomerID
CustomerName
Order:
  OrderTotal
OrderLines:
LineNo, ProductNo, Quantity, LinePrice
LineNo, ProductNo, Quantity, LinePrice
...
```

Data example

Native byte (character) stream:

```
1234
Boeing
1000
1,555,10,250.00;
2,666,10,750.00;

5678
Lockheed Martin
424
1,555,5,125.00;
2,777,1,100.00;
3,888,2,199.00;
```

Parsed values:

```
{OrdersArray=
  [ { CustomerName=Boeing, CustomerID=1234.0,
    Order=
      { OrderTotal=1000.0,
        OrderLines = [
          {OrderLine={LinePrice=250.0, ProductNo=555.0, Quantity=10.0,
            LineNo=1.0}},
          {OrderLine={LinePrice=750.0, ProductNo=666.0, Quantity=10.0,
            LineNo=2.0}}
        ]
      }
    },
  { CustomerName=Lockheed Martin, CustomerID=5678.0,
    Order=
      { OrderTotal=424.0,
        OrderLines = [
          { OrderLine={LinePrice=125.0, ProductNo=555.0, Quantity=5.0,
            LineNo=1.0}},
          { OrderLine={LinePrice=100.0, ProductNo=777.0, Quantity=1.0,
            LineNo=2.0}},
          { OrderLine={LinePrice=199.0, ProductNo=888.0, Quantity=2.0,
            LineNo=3.0}}
        ]
      }
    }
  ]
}
```

B.3.1.6 Data Padding

D3L supports data padding. Padding is unnamed gaps in a native format message that satisfy alignment constraints of the underlying native system. Padding is discarded in the Oracle Application Server Integration InterConnect application view message.

The following D3L example defines a number as a left-aligned string, which is padded at the end with blanks to a field width of 10.

```
<field name="Quantity">
  <number>
    <padstring length="10" padchar=' ' padstyle="tail" />
  </number>
```

The following native byte (character) stream satisfies this format:

```
9876.5_____
```

Pads can also be explicitly defined between fields in a structure by using the `<pad>` element.

The following D3L example shows two fields, which are separated by a pad of size 10.

```
<struct id="PROD">
  <field name="PRODID"> <termstring endchar=";" /> </field>
  <pad length="10" />
  <field name="PRODESC"> <termstring endchar=";" /> </field>
</struct>
```

The following native byte (character) stream would satisfy this format:

```
48682HW;~~~~~WASHER AND DRYER;
[...]
```

B.3.2 Comma-Separated Values File Parsing with D3L

A comma-separated values (CSV) file consists of multiple lines. Each line contains values separated by commas that end when a new line is required:

```
a,b,c,d
1,2,3
```

The string types, `termstring` and `simplestring`, have been added to parse CSV files.

- **termstring:** String type `termstring` is a variation of `limstring`. It requires only a terminating delimiter, but not a beginning delimiter. For example:

```
<termstring endchar="," />
```

This parses any string contents until encountering a comma.

- **simplestring:** String type `simplestring` is a special data type. It is used when the nearest parent structure defines a valid set of delimiters, which for the current data definition description language (D3L) library is limited to `limarray`. For example:

```
<limarray contchar="," endchar="\n">
  <simplestring />
</limarray>
```

The examples provided in the following sections use `imparray` so that input can be any number of elements, lines, or both.

B.3.2.1 CSVs are Assigned to Named Fields

This method assigns all CSVs on each line to named fields fixed number of fields per line. [Example B-17](#) describes CSVs assigned to named fields.

Example B–17 CSVs Assigned to Named Fields

```
<message name="createPhone" object="Phone" type="phoneRecord">
<impparray id="lines">
  <struct>
    <field name="rectype"> <termstring endchar="," /> </field>
    <field name="quantity"><termstring endchar="," /> </field>
    <field name="endHour"> <termstring endchar="," /> </field>
    <field name="endMin"> <termstring endchar="," /> </field>
    <field name="cost"> <termstring endchar="\n"/> </field>
  </struct>
</impparray>
<struct id="phoneRecord">
  <field name="csv"> <typeref type="lines" /> </field>
</struct>
</message>
```

The native format message payload for [Example B–17](#) is as follows:

```
4,,9,22,2324.29
''''
55,2342,11,46,728372339.57
```

B.3.2.2 All CSVs are Read into an Array

This method read all CSVs on each line into an array. There are variable number of fields per line. [Example B–18](#) describes all the CSVs read into an array.

Example B–18 All CSVs are Read into an Array

```
<message name="createPhone" object="Phone" type="phoneRecord">
<limarray id="linearr" contchar="#44" endchar="\n">
<simplestring />
</limarray>
<impparray id="myArray">
<struct>
<field name="line"> <typeref type="linearr" /> </field>
</struct>
</impparray>
<struct id="phoneRecord">
<field name="csv"> <typeref type="myArray" /> </field>
</struct>
</message>
```

The native format message for [Example B–18](#) is as follows:

```
4,,9,22,2324.29
55,2342,11,46,728372339.57
55,2342,11,46,728372339.57,4,,9,22,2324.29
1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0
```

B.3.2.3 Delimiter Encoding Styles

The delimiters for `limstring`, `termstring`, and `limarray` are enhanced to allow multiple characters, as well as additional encoding styles. The associated ASCII table codes are shown in parentheses:

- Escape code using `"\"`: This works for `"\r"` (13), `"\n"` (10), `"\t"` (9), and `"\f"` (12).

where:

- (13) is the ASCII code for a carriage return (CR)
- (10) is the ASCII code for a line feed (LF)
- (9) is the ASCII code for a horizontal tab (HT)
- (12) is the ASCII code for a form feed (FF)
- Escape ASCII code using "#": for example, "#13".
- Escape ASCII hexadecimal code using "#x": for example, "#x0D".
- End-of-file delimiter "\eof", which maps to a virtual end-of-file character: This delimiter can only be used once. No other fields can follow once it has been used.

[Example B-19](#) provides several examples of delimiter encoding styles.

Example B-19 Delimiter Encoding Styles

```
<termstring endchar="#x2C"/>
<termstring endchar="\n"/>
<limarray id="linearr" contchar="," endchar="\r\n">
  <simplestring/>
</limarray>

<termstring id="FileContents" endchar="\eof"/>
```

The "\r\n" on line 3 of [Example B-19](#) represents a DOS line break. The "\eof" on the last line of [Example B-19](#) represents an End of File.

B.4 D3L Integration with OracleAS Integration InterConnect Adapters

This section provides information on the D3L translation engine and its files' integration with the OracleAS Integration InterConnect adapter agent and bridge subcomponents to perform events and translations. It contains the following topics:

- [Runtime Initialization](#)
- [Native Format Message to Common View Incoming Message Translations](#)
- [Common View to Native Format Message Outgoing Messages Translations](#)

B.4.1 Runtime Initialization

The OracleAS Integration InterConnect Adapter agent reads .ini files (such as adapter.ini) at runtime to access each OracleAS Integration InterConnect adapter's configuration information. The OracleAS Integration InterConnect Adapter bridge are initialized and the configuration information provided by the OracleAS Integration InterConnect adapter agent. At the completion of a successful initialization, the OracleAS Integration InterConnect adapter bridge knows the following:

- The Oracle Application Server Integration InterConnect application name and its default endpoint (message destination)
- The various Oracle Application Server Integration InterConnect events to be handled by the OracleAS Integration InterConnect Adapter bridge
- D3L files that describe each of these events
- D3L files that are accessible and valid. If a file is invalid, then the OracleAS Integration InterConnect adapter cannot start

See Also:

- [Chapter 1, "Getting Started with OracleAS Integration InterConnect"](#)
- [Chapter 2, "Using iStudio"](#)

B.4.2 Native Format Message to Common View Incoming Message Translations

When the OracleAS Integration InterConnect adapter common transport layer detects an incoming message from an application, it receives the message in its native format. The message is then passed to the OracleAS Integration InterConnect adapter bridge. The bridge performs the following functions:

- Using the D3L translation engine to translate the native format message into an application view an Oracle Application Server Integration InterConnect message object.
- Raises an application view event

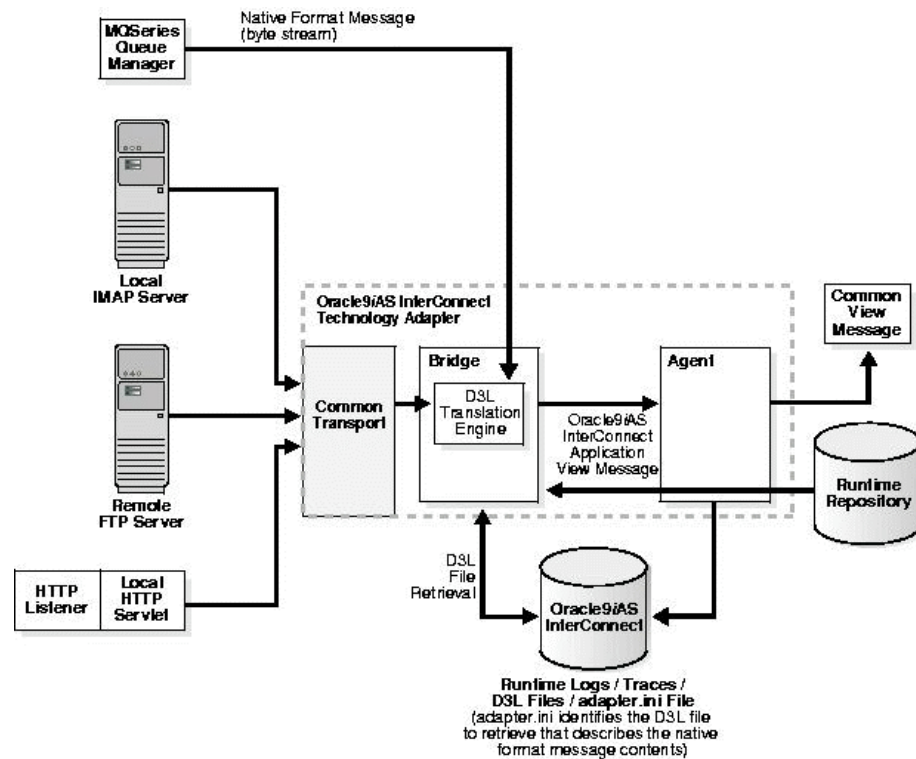
The agent transforms the application view event into a common view event and passes it on for further routing and processing. [Table B-1](#) describes the data flow sequence if D3L message header attributes are used.

Table B-1 *Message Header Attributes*

If The...	Then...
Name/value pair message header attributes are used	<p>The incoming native event might contain one of the following:</p> <ul style="list-style-type: none"> ■ Transport message headers/properties (made available to the bridge by the transport layer) ■ Transport message header parameter name (for example, <code>D3L-Header</code>) matches the header attribute of the message element in the D3L file (header="D3L-Header") ■ Transport message header value (for example, <code>D3L-Header: price</code>) matches the value attribute of the message element in the D3L file (value="price") <p>In the preceding cases, the bridge assumes that the matching D3L describes the incoming native event. Any conflicting header and value settings are detected and rejected by the bridge during initialization time.</p> <p>These operations are logged by OracleAS Integration InterConnect logging and tracing APIs for debugging, performance analysis, and business intelligence functions.</p>
Magic value message header attribute is used	<p>A magic value is specified by using the following:</p> <ul style="list-style-type: none"> ■ The D3L file (length = <i>n</i> bytes) ■ The first <i>n</i> bytes of payload data in an incoming native event (for example, <code>*UPDATE_PRICE</code>) match the magic attribute of the message element in the D3L file (for example, <code>magic="*UPDATE_PRICE"</code>) <p>In the preceding cases, the bridge assumes the native event must be processed using the matching D3L. If multiple D3Ls specify magic values that may match the same native event, the bridge randomly picks a D3L. This can lead to undesirable bridge behavior because the resulting application view event raised may not be the correct one.</p>

[Figure B-4](#) depicts the data flow sequence.

Figure B-4 Native Format Message to Common View Incoming Messages

**See Also:**

- ["Message Header Attributes"](#) on page B-6
- [Section B.6.4, "Task 4: Configure a Native Format Message with a D3L File"](#) on page B-28

B.4.3 Common View to Native Format Message Outgoing Messages Translations

When a common view event is raised, the OracleAS Integration InterConnect Adapter agent subscribing to the event, performs the following:

- Receives the associated message
- Transforms it to an Oracle Application Server Integration InterConnect message object
- Hands it to the OracleAS Integration InterConnect adapter bridge as an application view event

The OracleAS Integration InterConnect Adapter bridge queries the metadata associated with the event to determine the following:

- The D3L file for the D3L translation engine for the translation of the application view event into a native format message.
- The application to which the native format message event will be sent. There are two levels of rules to determine the application endpoint (the destination) of an OracleAS Integration InterConnect event:

- If the event contains metadata that specifies an endpoint, then the bridge uses this endpoint for the event. With the exception of the MQ Series adapter, all OracleAS Integration InterConnect adapters follow this rule.

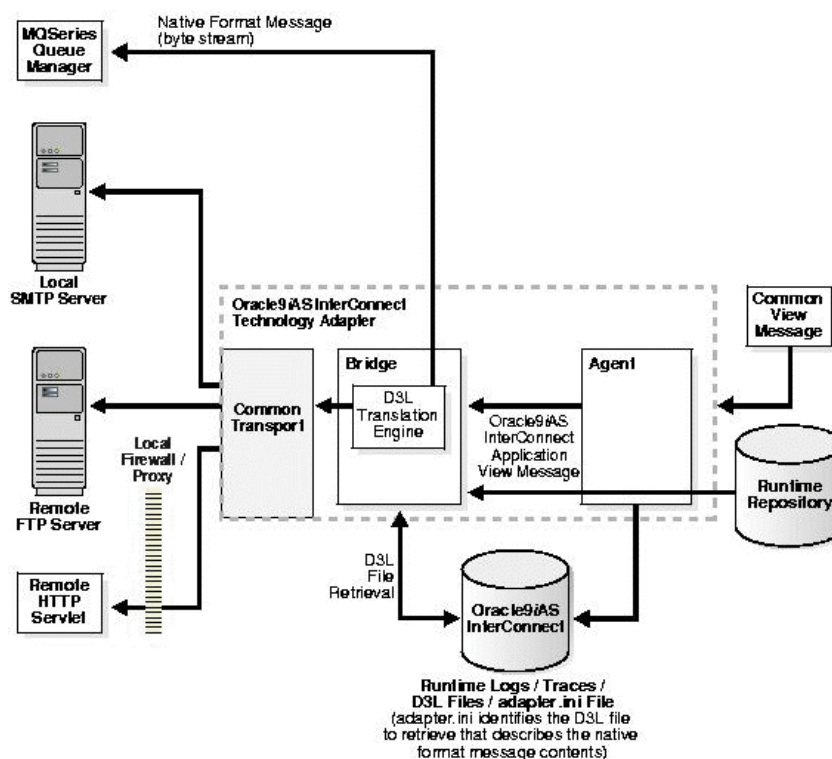
Note: Here the metadata itself names the endpoint and the content of the event is not searched.

- If the message metadata did not specify an endpoint, then the bridge uses its default endpoint, specified in the `adapter.ini` file, and made available to the bridge during initialization.

All OracleAS Integration InterConnect adapter operations are logged using the Oracle Application Server Integration InterConnect logging and tracing APIs for debugging, performance analysis, and other business intelligence functions.

Figure B-5 shows the data flow sequence.

Figure B-5 Common View to Native Format Message Outgoing Messages



B.5 Installing D3L

D3L is automatically installed with OracleAS Integration InterConnect.

See Also:

- *Oracle Application Server Installation Guide*
- *OracleAS Integration InterConnect Adapter documentation*

B.6 Configuring D3L

After installation, perform the following tasks to configure D3L:

- [Task 1: Configure D3L with iStudio](#)
- [Task 2: Create a Native Format Message](#)
- [Task 3: Create a D3L File Describing the Native Format Message](#)
- [Task 4: Configure a Native Format Message with a D3L File](#)
- [Task 5: Configure D3L with OracleAS Integration InterConnect Adapters](#)
- [Task 6: Import a D3L File in iStudio](#)
- [Task 7: Define Metadata Properties with Each Event \(Optional\)](#)

B.6.1 Task 1: Configure D3L with iStudio

You must define D3L in the `browsers.init` file. This enables you to import D3L files as attributes and select D3L as the message type in iStudio.

To integrate D3L with iStudio:

1. Use a text editor to open the

```
ORACLE_HOME\integration\interconnect\iStudio\browsers.init
file.
```

2. Add the following information at the end of the file:

```
D3L;oracle.oai.agent.adapter.technology.D3LBrowser;
```

3. Save your changes and exit the file.

See Also: ["Task 6: Import a D3L File in iStudio"](#) on page B-29

B.6.2 Task 2: Create a Native Format Message

The native format is typically predefined by your third-party application.

1. Create a native format message. For example, this native format message updates the salary of employee number 33201 to 55000:

```
*UPDATE_EMPLOYEE_SALARY* 33201 |55000|
```

Where...	Is...
UPDATE_EMPLOYEE_SALARY	action
33201	EmployeeID
55000	newSalary

B.6.3 Task 3: Create a D3L File Describing the Native Format Message

To create a D3L file (for example, `updemp.xml`) that describes the format of the native message:

1. Use a text editor. The following D3L file describes the contents of the native format message created in ["Task 2: Create a Native Format Message"](#).

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE message SYSTEM "d3l.dtd">
<message name="modify" object="Employee" type="modifyCommand">
```

```

header="D3L-Header" value="employee">
  <struct id="modifyCommand">
    <field name="action"><limstring delimiter="*" /></field>
    <field name="EmployeeID"><limstring delimiter=" " /></field>
    <field name="newSalary"><limstring delimiter="|" /></field>
  </struct>
</message>

```

2. Store the D3L file in the `ORACLE_`
`HOME\integration\interconnect\adapters\application` directory for direct access at deployment time.

See Also: The following sections for additional examples of D3L files:

- [Figure B-1](#) on page B-5
- [Example B-20](#) on page B-33
- ["Additional D3L Sample Files"](#) on page B-49

B.6.4 Task 4: Configure a Native Format Message with a D3L File

Configure a native format message with the correct D3L file. This enables the D3L translation engine to use the correct D3L file to verify native format message contents. For example, the D3L file created in ["Task 3: Create a D3L File Describing the Native Format Message"](#) includes settings for name/value pair message header attributes:

```

<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE message SYSTEM "d3l.dtd">
<message name="modify" object="Employee" type="modifyCommand"
  header="D3L-Header" value="employee">

```

These settings can match with the transport message header `D3L-Header` parameter name and `employee` value of a native format message:

```

POST /oai/servlet/transportServlet HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: acme.com:8888
Content-Length: 38
D3L-Header: employee

```

See Also:

- [Figure B-2](#) on page B-7
- [Table B-1](#) on page B-24

B.6.5 Task 5: Configure D3L with OracleAS Integration InterConnect Adapters

The `adapter.ini` file is read by the appropriate OracleAS Integration InterConnect Adapter at startup.

1. Use a text editor to open the

```

ORACLE_
HOME\integration\interconnect\adapters\application\adapter.ini

```

file.

In the preceding directory, `application` is the name of your application and the value of the `application` parameter in the `adapter.ini` file.

2. Ensure that parameter `ota.type` is set to the following value:


```
ota.type=D3L
```

This defines D3L as the message type for the OracleAS Integration InterConnect Adapter to handle incoming and outgoing messages.

3. Add the following line to define the D3L files for the bridge and D3L translation engine to use:

```
ota.d3ls=updemp.xml
```

where updemp.xml is the name of the D3L file created in "[Task 3: Create a D3L File Describing the Native Format Message](#)". Each event handled by the bridge must have its own D3L file. Whenever a new D3L file is imported in iStudio for use by an application, this parameter must be updated and the OracleAS Integration InterConnect adapter restarted.

4. Save your changes and exit the file.

B.6.6 Task 6: Import a D3L File in iStudio

iStudio enables you to import a D3L file for use with the following OracleAS Integration InterConnect features:

- Common data types
- Application data types
- Published/subscribed events
- Invoked/implemented procedures
- Business object events and procedures

When a D3L file is associated with Oracle Application Server Integration InterConnect common data types, application data types, events, or procedures, an iStudio OracleAS Integration InterConnect Adapter browser plug-in verifies that the file conforms to the syntax and semantics of D3L. [Table B-2](#) identifies the iStudio tasks and locations where you can import a D3L file as an attribute and select D3L as a message type. Documentation references that describe how to perform these tasks are also provided.

Table B-2 D3L Functionality in iStudio

For this D3L Functionality...	Do This...
Common Data Type Tasks:	
<ul style="list-style-type: none"> ■ Create a common data type that imports a D3L file as an attribute. 	Refer to " Creating Common Data Types " on page 3-2
Application Data Types Tasks:	
<ul style="list-style-type: none"> ■ Creating an application data type that imports a D3L file as an attribute. 	Select New from File, then select Application Data Type from the iStudio menu
Event Tasks:	
<ul style="list-style-type: none"> ■ Create an event that imports a D3L file as an attribute. 	Refer to " Creating Events " on page 4-2
<ul style="list-style-type: none"> ■ Publish an event that uses D3L as the message type and imports a D3L file as an attribute. 	Refer to " Publishing an Event " on page 4-3

Table B-2 (Cont.) D3L Functionality in iStudio

For this D3L Functionality...	Do This...
<ul style="list-style-type: none"> Subscribe to an event that uses D3L as the message type and imports a D3L file as an attribute. 	Refer to " Subscribing to an Event " on page 4-8
Procedure Tasks:	
<ul style="list-style-type: none"> Create a procedure that imports a D3L file as an attribute. 	Refer to " Creating a Procedure " on page 5-2
<ul style="list-style-type: none"> Invoke a procedure that uses D3L as the message type and imports a D3L file as an attribute. 	Refer to " Invoking a Procedure " on page 5-3
<ul style="list-style-type: none"> Implement a procedure that uses D3L as the message type and imports a D3L file as an attribute. 	Refer to " Implementing a Procedure " on page 5-6

Note: D3L functionality with procedures in iStudio is only available with the MQ Series adapter.

B.6.7 Task 7: Define Metadata Properties with Each Event (Optional)

You can associate metadata with each event in iStudio by selecting Modify Fields in the Subscribe Wizard - Define Application View. The Modify Fields appears after you select D3L as the Message Type in the preceding Subscribe Wizard - Select an Event. Such metadata is used for content-based routing of events at runtime.

The following application view event metadata is used by the OracleAS Integration InterConnect adapters. The property name is prefixed by `ota` to minimize namespace conflicts with user-defined metadata on application view events. The property name is considered a keyword/reserved name, and is used by both iStudio and the bridge, and must be kept consistent between these two components.

Property Name	Property Value Type	Explanation
<code>ota.d3lPath</code>	The D3L filename (string). This is automatically set. Do not modify this property.	The path name (relative or absolute) of the file that contains the D3L guidelines for this event.
<code>ota.isD3L</code>	This value is always true (boolean) and automatically set. Do not modify this property.	A flag indicating that this event is based on D3L.
<code>ota.send.endpoint</code>	The endpoint URL (string). This is mandatory. For example: <code>http://foo.com/servlet/test</code>	The actual endpoint to which this message is sent. This setting must match the type of OracleAS Integration InterConnect Adapter that subscribes to the event.

Property Name	Property Value Type	Explanation
http.sender.* file.sender.*	Refer to Chapter 2 of the appropriate OracleAS Integration InterConnect Adapter documentation for the adapter being defined in the ota.send.endpoint parameter URL ¹). This is optional. For example: http.sender.timeout=500 0	The properties define the transport layer configuration.

¹ The MQ Series adapter does not define any smtp.sender properties. This is because the MQ Series adapter does not support multiple sending endpoints in this release.

See Also: "Subscribing to an Event" on page 4-8

B.7 D3L Use Case

This section contains these topics:

- [D3L Use Case Overview](#)
- [Creating Data Type Definitions for Application Views](#)
- [Configuring the aqapp_pub and fileapp_sub Applications in iStudio](#)
- [Installing the Advanced Queuing and FTP Adapters](#)
- [Running the D3L Use Case](#)
- [Using Other Adapters in D3L and XML Modes](#)

B.7.1 D3L Use Case Overview

This use case provides an example of a minimal Oracle Application Server Integration InterConnect configuration and setup that uses D3L. This use case involves two applications using OracleAS Integration InterConnect Adapters:

- aqapp_pub, which is based on the Advanced Queuing adapter
- fileapp_sub, which is based on the FTP adapter running in D3L mode

These applications use a business object called `Employee`, which has one defined event called `newEmployee`.

The `aqapp_pub` application publishes the `newEmployee` event, while `fileapp_sub` subscribes to it. [Table B-3](#) describes the attributes (message structure) of the `newEmployee` event:

Table B-3 Common View Attributes

Attribute Name	Attribute Type
EmpName	String
EmpDept	Integer
EmpHiredate	Date
EmpSalary	Double

All these attributes are scalar (that is, there are no arrays). This message structure represents the common view of the `newEmployee` event. For simplicity, the application views for the two applications have the exact same structure as the common view.

In "Creating Data Type Definitions for Application Views", a DTD file and a D3L file are created that match the common view attributes shown in [Table B-3](#) on page B-31. These files are used when the application views for the two applications are defined.

B.7.2 Creating Data Type Definitions for Application Views

You must create data type definitions for the two application views.

This section contains these topics:

- [Task 1: Create a DTD File for the Advanced Queuing Adapter](#)
- [Task 2: Create a D3L File for the FTP Adapter](#)

Note: This use case assumes that you have already installed and configured Oracle Application Server Integration InterConnect and iStudio.

B.7.2.1 Task 1: Create a DTD File for the Advanced Queuing Adapter

The application view for the Advanced Queuing adapter must be defined through a DTD. The DTD enables the Advanced Queuing adapter to translate a received XML (text) document into a runtime application view (Java) object. The agent component of the Advanced Queuing adapter can then transform it to a common view object before routing it to any application subscribers. A DTD is registered with (imported to) the application while defining, for example, a publication in iStudio.

1. Create a DTD file that matches the common view message structure shown in [Table B-3](#) on page B-31:

```
<!ELEMENT NewEmpRec (EmpName, EmpDept, EmpHiredate, EmpSalary)>
<!ELEMENT EmpName    (#PCDATA)>
<!ELEMENT EmpDept    (#PCDATA)>
<!ELEMENT EmpHiredate (#PCDATA)>
<!ELEMENT EmpSalary  (#PCDATA)>
```

2. Save this DTD in a text file named `newemp.dtd`. This file can be saved to any location.

B.7.2.2 Task 2: Create a D3L File for the FTP Adapter

When running in D3L mode, the FTP adapter must have its application view defined by a D3L (XML) file. The D3L file enables a bidirectional translation between the internal runtime application view (Java) object representation and an external binary/native format message representation. The D3L file is registered with (imported to) the application while defining, for example, a subscription in iStudio.

Assume the external binary native format message of the `newEmployee` event is as follows:

```
message ::= <empname> <empdept> <emphiredate> <empsalary>
empname ::= char[20] // left adjusted string, 20 chars wide, right padded with
spaces
empdept ::= byte[2]  // unsigned 2-byte integer, little endian
emphiredate ::= '|' + <month> + <anysep> + <day> + <anysep> + <year> + '|''
```

```
empsalary ::= '$' <number> '$'
```

Where...	Is...
<month>, <day>, and <year>	The date format elements MM, DD, and YYYY (all digits)
<anysep>	Any single character
<number>	Any decimal number using the character "." as a decimal separator

1. Create a D3L file that describes the structure that the native format message must follow to communicate with Oracle Application Server Integration InterConnect. The native format message can be expressed/mapped in the D3L XML definition as shown in [Example B-20](#):

Example B-20 D3L Sample File

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE message SYSTEM "d3l.dtd">
<message type="NewEmpRec" name="newEmployee" object="Employee">
  <!-- TYPE DECLARATIONS -->
  <!-- string field 20 chars wide with trailing spaces -->
  <padstring id="str20" padchar=" " padstyle="tail" length="20" />
  <!-- unsigned 2-byte integer -->
  <unsigned2 id="uword" endian="little" />
  <!-- date format using pattern MM-DD-YYYY enclosed by '|' -->
  <date id="date" format="MMDDYYYY"><limstring delimiter="|" />
  </date>
  <!-- decimal number format enclosed by '$' -->
  <number id="number"><limstring delimiter="$" /></number>
<!-- MESSAGE STRUCTURE -->
<struct id="NewEmpRec">
  <field name="EmpName"> <typeref type="str20" /> </field>
  <field name="EmpDept"> <typeref type="uword" /> </field>
  <field name="EmpHiredate"> <typeref type="date" /> </field>
  <field name="EmpSalary"> <typeref type="number" /> </field>
</struct>
</message>
```

2. Save the D3L definition in [Example B-20](#) to a file called `newemp.xml`.
3. Include a copy of this file on both the host computer where Oracle Application Server Integration InterConnect is installed and on the Windows computer where iStudio is installed.

Note: `newemp.xml` is also copied to the FTP adapter application directory in "[Task 4: Copy the newemp.xml D3L File to the fileapp_sub Adapter Directory](#)" on page B-44.

The following example shows a native format message that can be translated by the `newemp.xml` D3L file (The ? character means nonprintable):

Pos	Bytes (in hexadecimal)	Characters
0000000	4a6f 686e 2044 6f65 2020 2020 2020 2020	John Doe
0000020	2020 2020 4000 7c31 322f 3134 2f32 3030	@? 12/14/200
0000040	317c 2435 3432 3230 2e37 3524	1 \$54220.75\$

Where...	Is...
EmpName	John Doe
EmpDept	64 (hex: 0x40)
EmpHiredate	12/14/2001
EmpSalary	54220.75

In "[Configuring the aqapp_pub and fileapp_sub Applications in iStudio](#)" on page B-34, you complete all the steps necessary in iStudio, including defining the common view, defining the application creation, and so on.

B.7.3 Configuring the aqapp_pub and fileapp_sub Applications in iStudio

This section describes the tasks to complete in iStudio.

This section contains these topics:

- [Task 1: Create a New Workspace and New Project](#)
- [Task 2: Create the Employee Business Object](#)
- [Task 3: Create the newEmployee Event](#)
- [Task 4: Create the aqapp_pub Application](#)
- [Task 5: Enable the aqapp_pub Application to Publish the newEmployee Event](#)
- [Task 6: Define the Application Queue for the aqapp_pub Application](#)
- [Task 7: Create the fileapp_sub Application](#)
- [Task 8: Enable the fileapp_sub Application to Subscribe to the newEmployee Event](#)

B.7.3.1 Task 1: Create a New Workspace and New Project

1. Start iStudio from the Start menu.

When iStudio starts, the last used workspace is automatically loaded. For this use case, define a new workspace and new Project.
2. Select **File**, and then **New Workspace**.
3. Enter `d3l_tests` for the Workspace Name, and click **OK**.
4. Select **File**, and then **New Project**.
5. Enter `d3l_test_ftp` for the Project Name, and click **OK**.
6. Enter the following values in the Hub Information dialog:

For...	Enter...
Hub database username	<code>oaihub</code>
Hub database password	<code>oaihub</code> (the default)
Hub database URL	<code>hubDB-host:hubDB-port:hubDB-SID</code> For example: <code>dlsun10:1521:V904</code>

B.7.3.2 Task 2: Create the Employee Business Object

1. Select **File, New**, and then **Business Object**.
2. Enter `Employee` for the Business Object name and click **OK**.

Note: The `Employee` Business Object name matches with the value for the `object` attribute of the `<message>` element in the D3L file created in "[Task 2: Create a D3L File for the FTP Adapter](#)" on page B-32.

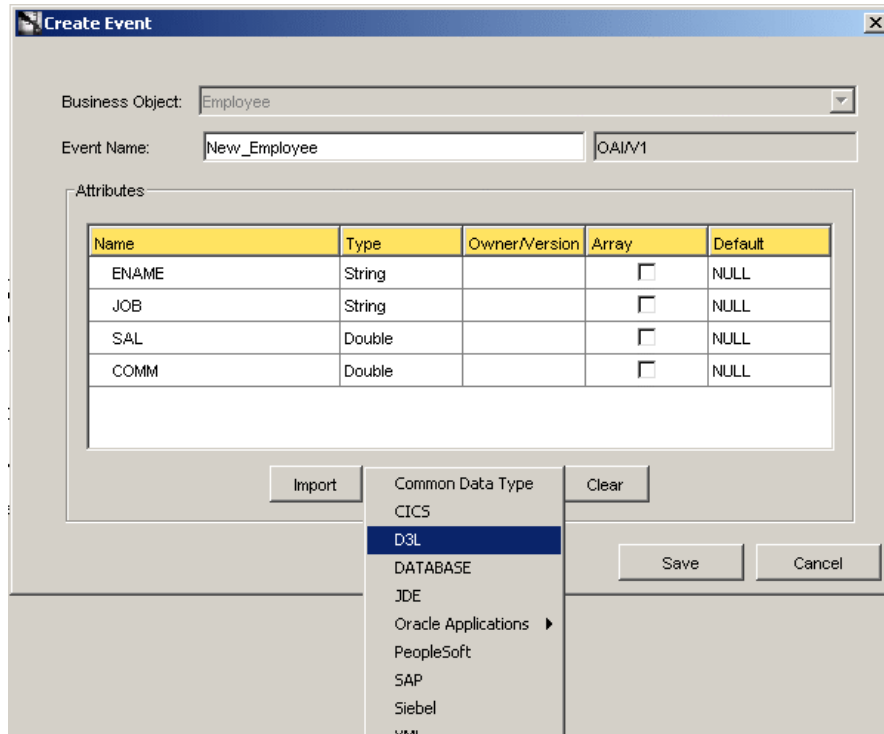
B.7.3.3 Task 3: Create the newEmployee Event

Define the `newEmployee` event as described in "[D3L Use Case Overview](#)" on page B-31. Define the (common view) attributes of the event by importing the `newemp.xml` D3L file defined in "[Task 2: Create a D3L File for the FTP Adapter](#)" on page B-32. This D3L file defines the same data types as used by the common view. (Refer to [Table B-3](#) on page B-31.)

1. Select **File, New**, and then **Event**.
2. Select `Employee` in the Business Object list.
3. Enter `newEmployee` in the Event Name field.
4. Click **Import**.
5. Select D3L from the list that appears.
6. Locate and select the `newemp.xml` D3L file created in "[Task 2: Create a D3L File for the FTP Adapter](#)" on page B-32. The contents of `newemp.xml` display in the Attributes fields of the Create Event dialog. If you receive an error while importing, check if the contents of the `newemp.xml` file on your iStudio computer are identical to the text shown in [Example B-20](#) on page B-33.

Note: The `newEmployee` Event Name matches with the value for the `name` attribute of the `<message>` element in the D3L file created in "[Task 2: Create a D3L File for the FTP Adapter](#)" on page B-32.

The Create Event dialog is displayed as follows:



7. Click **Save**.

See Also: ["Creating Events"](#) on page 4-2

B.7.3.4 Task 4: Create the aqapp_pub Application

Now create the aqapp_pub application, which publishes the defined event Employee.newEmployee.

1. Click **File, New**, and then **Application**.
2. Enter aqapp_pub for the Application Name, and click **OK**.

B.7.3.5 Task 5: Enable the aqapp_pub Application to Publish the newEmployee Event

Use the Publish Wizard to publish the newEmployee event.

This section contains the following topics:

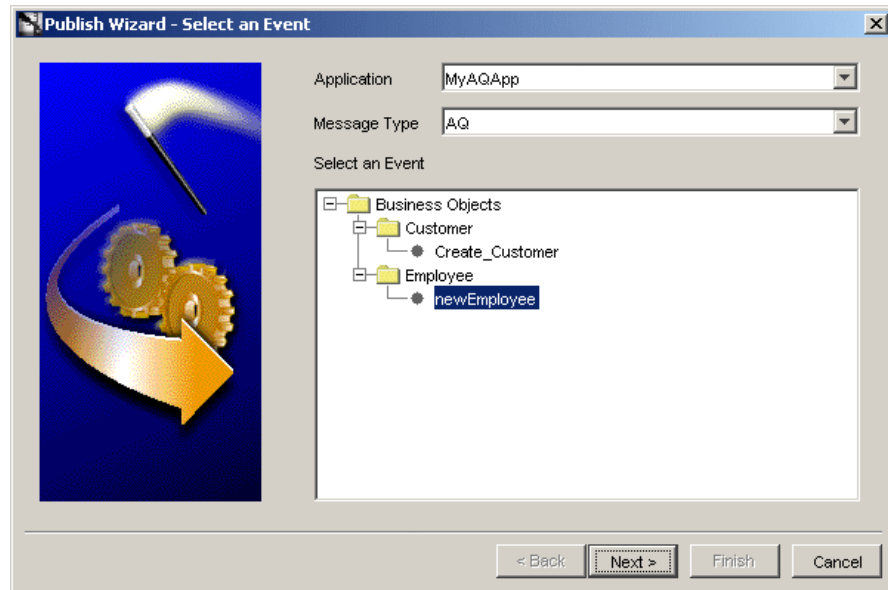
- [Select the Event to Publish](#)
- [Define the Application View](#)
- [Define the Application View to Common View Mapping](#)

B.7.3.5.1 Select the Event to Publish

Select the event to publish with the Publish Wizard.

1. Select **Event**, and then **Publish Event**. The Publish Wizard - Select an Event dialog is displayed.
2. Select aqapp_pub from the Application list.

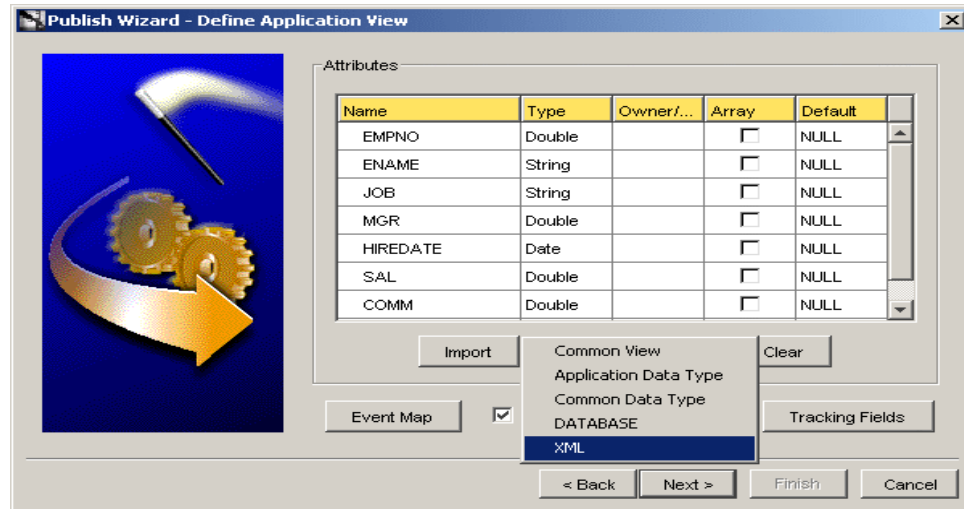
3. Select AQ from the Message Type list. This choice means that the `aqapp_pub` application is based on the Advanced Queuing adapter.
4. Click the `newEmployee` event in the Select an Event list, which is a child of the `Employee` business object.



5. Click **Next**. The Publish Wizard - Define Application View dialog is displayed.

B.7.3.5.2 Define the Application View Define the application view for the Advanced Queuing adapter-based application `aqapp_pub` in this dialog. This view was defined in "[Task 1: Create a DTD File for the Advanced Queuing Adapter](#)" on page B-32 as an XML DTD, which is a requirement of the Advanced Queuing adapter. Import this DTD to define the application view.

1. Click **Import**.
2. Select XML from the list that appears.
3. Locate and select the `newemp.dtd` file, which you created in "[Task 1: Create a DTD File for the Advanced Queuing Adapter](#)" on page B-32.
4. Select `NewEmpRec` in the Choose Root Element dialog.
5. Click **OK**. The Publish Wizard - Define Application View dialog is displayed.

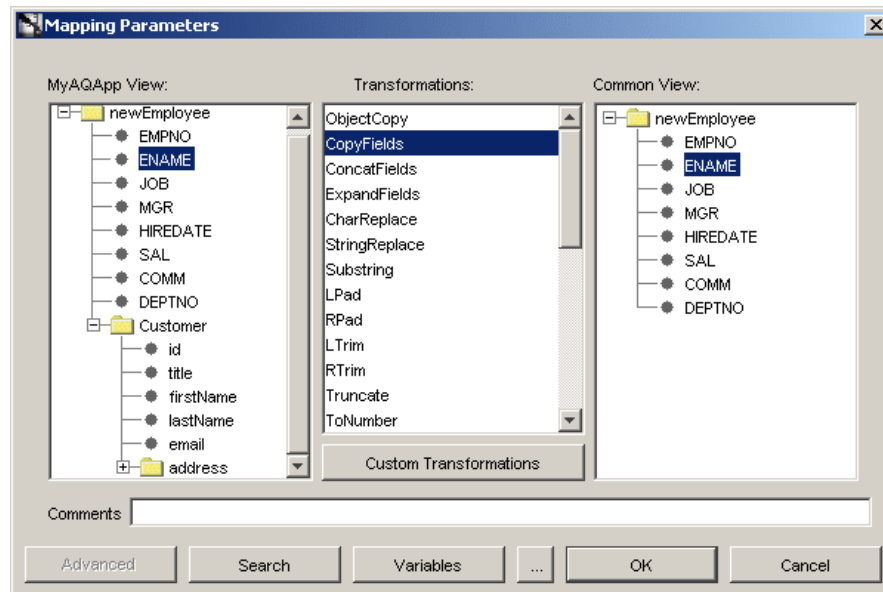


- Click **Next**. The Publish Wizard - Define Mapping dialog is displayed.

B.7.3.5.3 Define the Application View to Common View Mapping

Define the application view to common view mapping on this dialog.

- Click **New**. The Mapping Parameters dialog is displayed.
- Expand `newEmployee` and `NewEmpRec` (clicking the '+') in the `aqapp_pub` View pane.
- Expand `newEmployee` and `NewEmpRec` (clicking the '+') in the Common View pane.
- Click the `EmpName` attribute in both panes.
- Select `CopyFields` in the Transformations list. The Mapping Parameters dialog is displayed as follows:



- Click **OK**.

7. Repeat Steps 4 through 6 for the remaining attributes `EmpDept`, `EmpHiredate`, and `EmpSalary`. When complete, the Publish Wizard - Define Mapping dialog is displayed.
8. Click **Finish**. The Publication for application `aqapp_pub` is complete. The navigation pane on the left hand side of iStudio shows the structure for the `aqapp_pub` application.

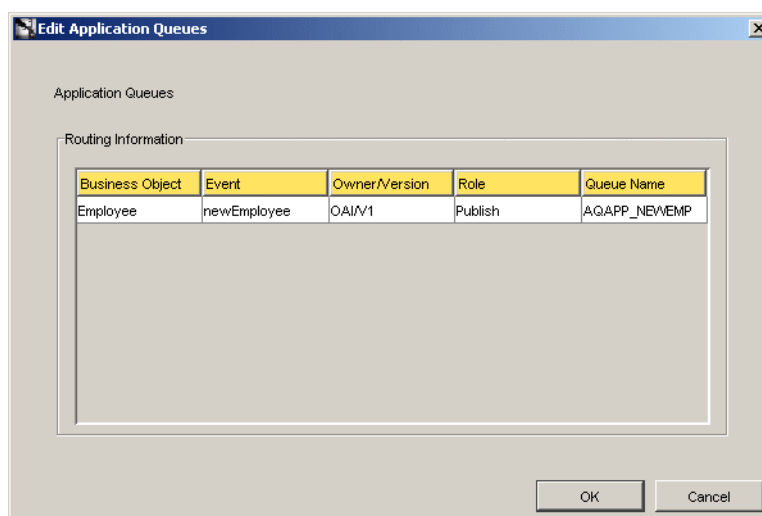
B.7.3.6 Task 6: Define the Application Queue for the `aqapp_pub` Application

As the `aqapp_pub` application publishes the `newEmployee` event and is based on the Advanced Queuing adapter, you must define the (Oracle Advanced Queuing) queue from which the Advanced Queuing adapter reads the event. When an XML message, which complies with the DTD defined in ["Task 1: Create a DTD File for the Advanced Queuing Adapter"](#) on page B-32, is enqueued onto the outbound queue, the Advanced Queuing adapter:

- Picks the message up
- Translates the message to an application view event
- Passes the message to the adapter agent for further transformation to the common view representation

The following steps describe how to choose the queue name. The queue does not have to exist physically at this point, as you create it in a later step. (Refer to section ["Task 2: Create the Application Queue AQAPP_NEWEMP"](#) on page B-42.)

1. Click the **Deploy** navigation tab on top of the iStudio navigation list.
2. Expand the `Applications` node.
3. Expand the `aqapp_pub` node.
4. Expand the `Routing` node.
5. Right-click the `Application Queues` node.
6. Select the `Edit` option from the list that appears. The `Edit Application Queues` dialog is displayed.
7. Click in the empty field under the `Queue Name` column header, and enter the chosen queue name, for example, `AQAPP_NEWEMP`:



8. Click **OK**.

B.7.3.7 Task 7: Create the fileapp_sub Application

Create the `fileapp_sub` application to subscribe to the defined event `Employee.newEmployee` (which is published by `aqapp_pub`).

1. Click **File, New**, and then **Application**.
2. Enter `fileapp_sub` for the Application Name and click **OK**.

B.7.3.8 Task 8: Enable the fileapp_sub Application to Subscribe to the newEmployee Event

Use the Subscribe Wizard to subscribe to the `newEmployee` event.

This section contains these topics:

- [Select the Event to which to Subscribe](#)
- [Define the Application View](#)
- [Define the Application View to Common View Mapping](#)

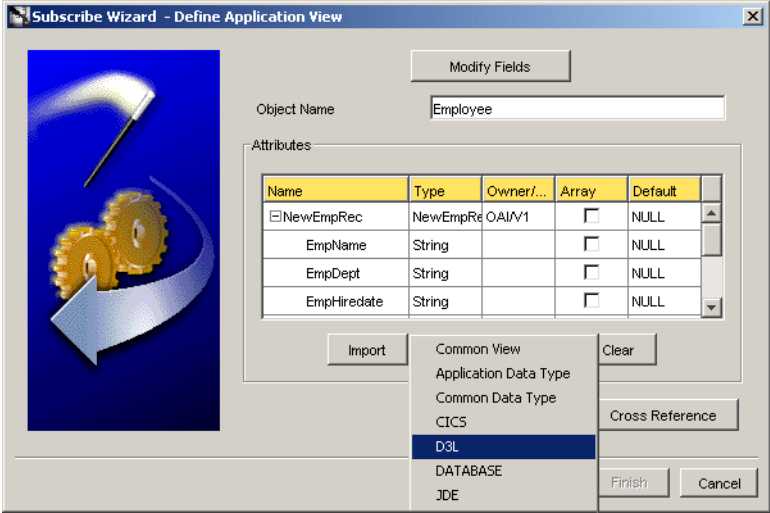
B.7.3.8.1 Select the Event to which to Subscribe Select the event to which to subscribe with the Subscribe Wizard.

1. Select **Event**, and then **Subscribe Event**. The Subscribe Wizard - Select an Event dialog is displayed.
2. Select `fileapp_sub` from the Application list.
3. Select `D3L` from the Message Type list.
4. Click `newEmployee` (under `Employee`) in the Select an Event list.
5. Click **Next**. The Subscribe Wizard - Define Application View dialog appears.

B.7.3.8.2 Define the Application View Define the application view for the FTP adapter-based application `fileapp_sub` in this dialog. This view was defined in "[Task 2: Create a D3L File for the FTP Adapter](#)" on page B-32 as a D3L file. This is a requirement of any OracleAS Integration InterConnect Adapter operating in D3L mode. Import this D3L file to define the application view.

1. Enter `Employee` as the business object name in the Object Name input field.
2. Click **Import**.
3. Select `D3L` from the list that appears.
4. Locate and select the `newemp.xml` file, which you saved in "[Task 2: Create a D3L File for the FTP Adapter](#)" on page B-32.

The contents of `newemp.xml` display in the Attributes fields:



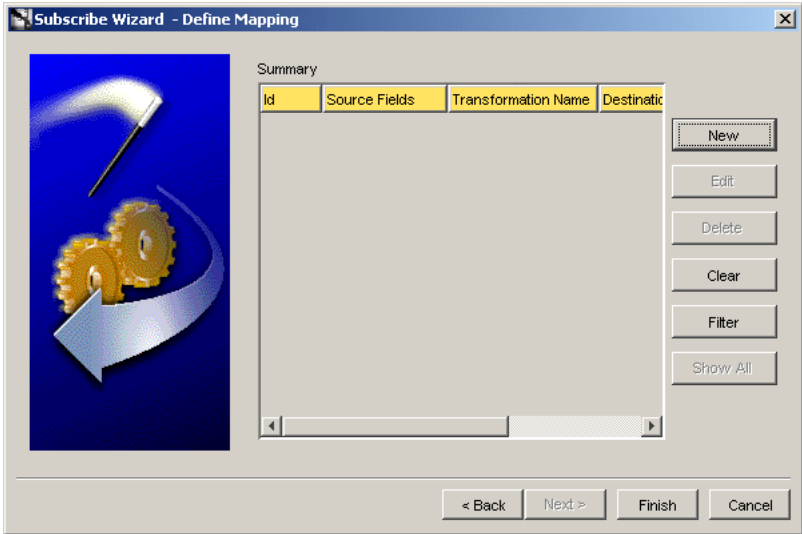
5. Click **Next**. The Subscribe Wizard - Define Application View dialog appears.

B.7.3.8.3 Define the Application View to Common View Mapping Define the application view to common view mapping in this dialog.

1. Click **New**. The Mapping Parameters dialog appears.
2. Expand newEmployee by clicking the '+' in the Common View pane.
3. Expand newEmployee by clicking the '+' in the fileapp_sub View pane.
4. Click the NewEmpRec node in both panes.
5. Select ObjectCopy in the Transformations list and click **OK**.

Note: You can choose ObjectCopy here because the common view and application view are based on the same D3L file.

The Subscribe Wizard - Define Mapping dialog appears as follows:



6. Click **Finish**. This completes the necessary setup steps in iStudio.

B.7.4 Installing the Advanced Queuing and FTP Adapters

Now that iStudio setup is complete, you must install one instance of each of the two adapter types. This section contains these topics:

- [Task 1: Install the Advanced Queuing Adapter for Application aqapp_pub](#)
- [Task 2: Create the Application Queue AQAPP_NEWEMP](#)
- [Task 3: Install the FTP Adapter for Application fileapp_sub](#)
- [Task 4: Copy the newemp.xml D3L File to the fileapp_sub Adapter Directory](#)
- [Task 5: Set the D3L file and Payload Type in the adapter.ini Adapter Initialization File](#)

B.7.4.1 Task 1: Install the Advanced Queuing Adapter for Application aqapp_pub

1. Refer to "Advanced Queuing Adapter Installation" in Chapter 2 of the *Oracle Application Server Integration InterConnect Adapter for AQ Installation and User's Guide* for installation instructions.

During installation, enter the following specific values when prompted:

- a. Enter `aqapp_pub` in the Application Name field of the Oracle Application Server Integration InterConnect AQ Adapter Configuration dialog.
 - b. Enter the database connection information to connect to the database instance on the Application Spoke Database page. The `AQAPP_NEWEMP` application queue defined in "[Task 6: Define the Application Queue for the aqapp_pub Application](#)" on page B-39 is created here.
 - c. Enter the database username and password of the account and schema on the Spoke Application Database AQ Username dialog, which owns the Application Queue (`AQAPP_NEWEMP`). Select the schema name `aqapp` and the password `aqapp`. Leave the Consumer Name field blank, as you are creating the `AQAPP_NEWEMP` queue as a single consumer queue.
2. Complete adapter installation by providing appropriate responses when prompted.

When installation is complete, the new adapter instance is located in the following directory:

Platform	Directory
Windows	%ORACLE_ HOME%\integration\interconnect\adapters\aqapp_ pub
UNIX	\$ORACLE_ HOME/integration/interconnect/adapters/aqapp_ pub

B.7.4.2 Task 2: Create the Application Queue AQAPP_NEWEMP

To create the Advanced Queuing `AQAPP_NEWEMP` application queue, you must first create the queue table, create the queue, and start the queue.

1. Ensure that the database user issuing the commands in this section has been granted the following roles:

```
RESOURCE,
CONNECT,
AQ_ADMINISTRATOR_ROLE
```

2. Use SQL*Plus to log in to the database account specified in Step 1 of "[Task 1: Install the Advanced Queuing Adapter for Application aqapp_pub](#)" on page B-42.

3. Create the queue table using the same name as the application queue:

```
SQL> EXECUTE dbms_aqadm.create_queue_table('AQAPP_NEWEMP', 'RAW');
```

4. Create the queue:

```
SQL> EXECUTE dbms_aqadm.create_queue('AQAPP_NEWEMP', 'AQAPP_NEWEMP');
```

5. Start the queue:

```
SQL> EXECUTE dbms_aqadm.start_queue('AQAPP_NEWEMP');
```

B.7.4.3 Task 3: Install the FTP Adapter for Application fileapp_sub

1. Refer to "FTP Adapter Installation" in Chapter 2 of the *Oracle Application Server Integration InterConnect Adapter for FTP Installation and User's Guide* for installation instructions.

During installation, enter the following specific values when prompted:

- a. Enter fileapp_sub in the Application Name field of the Oracle Application Server Integration InterConnect FTP Adapter Configuration dialog.
- b. Enter the following value in the URL field of the Oracle Application Server Integration InterConnect FTP Adapter Configuration Configure receiving endpoint information dialog:

```
ftp://localhost/tmp/fileapp_sub/read
```

- c. Enter the following value in the URL field of the Oracle Application Server Integration InterConnect FTP Adapter Configuration Configure sending endpoint information dialog:

```
ftp://localhost/tmp/fileapp_sub/write
```

This action places every newEmployee message received by the fileapp_sub application (by way of its configured subscription created in "[Task 8: Enable the fileapp_sub Application to Subscribe to the newEmployee Event](#)" on page B-40) in the /tmp/fileapp_sub/write directory of the computer where the FTP adapter is installed. Ensure that you create these directories with global read and write permissions before starting the fileapp_sub application (based on the FTP adapter), for example:

```
$ umask 0
$ mkdir -p /tmp/fileapp_sub/read
$ mkdir -p /tmp/fileapp_sub/write
```

2. Complete adapter installation by providing appropriate responses when prompted.

When installation is complete, the new adapter instance is located in the following directory:

Platform	Directory
Windows	%ORACLE_ HOME%\integration\interconnect\adapters\fileapp_ _sub
UNIX	\$ORACLE_ HOME/integration/interconnect/adapters/fileapp_ sub

B.7.4.4 Task 4: Copy the newemp.xml D3L File to the fileapp_sub Adapter Directory

1. Copy the newemp.xml D3L file defined in "[Task 2: Create a D3L File for the FTP Adapter](#)" on page B-32 to the platform-specific directory mentioned in the preceding Step 2 on page B-43.

B.7.4.5 Task 5: Set the D3L file and Payload Type in the adapter.ini Adapter Initialization File

Set the ota.d3ls and ota.type parameters in the adapter.ini adapter initialization file for the FTP adapter. The adapter.ini file is located in the platform-specific directory mentioned in the preceding Step 2 on page B-43.

1. Use a text editor to set the ota.d3ls parameter to newemp.xml in adapter.ini:

```
ota.d3ls=newemp.xml
```

If the ota.d3ls parameter line already exists in adapter.ini, replace it with the preceding version.

2. Use a text editor to set the ota.type parameter to D3L in adapter.ini:

```
ota.type=D3L
```

B.7.5 Running the D3L Use Case

Now that both the Advanced Queuing adapter instance aqapp_pub and the FTP adapter instance fileapp_sub have been installed, use both to run the D3L use case.

This section contains these topics:

- [Task 1: Start the Adapters](#)
- [Task 2: Create PL/SQL Code to Trigger the Native newEmployee Event](#)
- [Task 3: Trigger the newEmployee Event](#)
- [Task 4: Verify Receipt of newEmployee Event](#)

B.7.5.1 Task 1: Start the Adapters

The first task is to start the adapters.

B.7.5.1.1 To Start the Adapters on UNIX: Follow these steps to start the adapters on UNIX:

To start the aqapp_pub (Advanced Queuing) adapter:

1. Change directories to where the aqapp_pub adapter is installed:

```
$ cd $ORACLE_HOME/integration/interconnect/adapters/aqapp_pub
```


2. Start the adapter as a background process:

```
$ start &
```

To start the `fileapp_sub` (FTP) adapter:

1. Change directories to where the `fileapp_sub` adapter is installed:

```
$ cd $ORACLE_HOME/integration/interconnect/adapters/fileapp_sub
```

2. Start the adapter as a background process:

```
$ start &
```

B.7.5.1.2 To Start the Adapters on Windows: Follow these steps to start the adapters on Windows:

To start the `aqapp_pub` (Advanced Queuing) adapter:

1. Change directories to where the `aqapp_pub` adapter is installed:

```
cd %ORACLE_HOME%\integration\interconnect\adapters\aqapp_pub
```

2. Start the adapter:

```
start
```

To start the `fileapp_sub` (FTP) adapter:

1. Change directories to where the `fileapp_sub` adapter is installed:

```
cd %ORACLE_HOME%\integration\interconnect\adapters\fileapp_sub
```

2. Start the adapter:

```
start
```

Note: You can also start adapters from the Windows Control Panel. Refer to the OracleAS Integration InterConnect Adapter documentation for instructions.

B.7.5.2 Task 2: Create PL/SQL Code to Trigger the Native `newEmployee` Event

The next task generates the native event (that is, triggers the `newEmployee` event). As configured in iStudio, the `aqapp_pub` application publishes the `newEmployee` event. It does so when it sees a new (XML) message on the `AQAPP_NEWEMP` queue that conforms to the DTD defined in "[Task 1: Create a DTD File for the Advanced Queuing Adapter](#)" on page B-32.

To generate the native event, you must enqueue a message on the application queue (`AQAPP_NEWEMP`) for the application `aqapp_pub`. You do this through an anonymous PL/SQL block.

1. Change directories to where the `aqapp_pub` application (of the Advanced Queuing adapter) is installed, for example:

On...	Change To...
UNIX	<code>\$ cd \$ORACLE_HOME/integration/interconnect/adapters/aqapp_pub</code>

On...	Change To...
Windows	cd %ORACLE_ HOME%\integration\interconnect\adapters\aqapp_pub

2. Create a file (named `newemp.sql` in this example) with the contents shown in [Example B-21](#):

Example B-21 File `newemp.sql`

```

DECLARE
    enqueue_options    dbms_aq.enqueue_options_t;
    message_properties dbms_aq.message_properties_t;
    msgid              RAW(16);
    raw_payload        RAW(32767);
    payload            varchar2(2000);
BEGIN
    payload :=
        '<?xml version="1.0" standalone="no"?>
        <NewEmpRec>
            <EmpName>Scott Tiger</EmpName>
            <EmpDept>257</EmpDept>
            <EmpHiredate>05/01/2001</EmpHiredate>
            <EmpSalary>52308.75</EmpSalary>
        </NewEmpRec>';

    raw_payload := utl_raw.cast_to_raw(payload);
    dbms_aq.enqueue(queue_name      => 'AQAPP_NEWEMP',
                    enqueue_options => enqueue_options,
                    message_properties => message_properties,
                    payload          => raw_payload,
                    msgid            => msgid);

    commit;
END;
/

```

Note: The `payload` variable is assigned a string value, which contains a valid XML document that conforms to the DTD `newemp.dtd` defined in ["Task 1: Create a DTD File for the Advanced Queuing Adapter"](#) on page B-32.

B.7.5.3 Task 3: Trigger the newEmployee Event

Everything is now defined, created, and started. You must now trigger the `newEmployee` event, which was prepared in ["Task 2: Create PL/SQL Code to Trigger the Native newEmployee Event"](#) on page B-45.

As mentioned earlier, the event is triggered when you place an XML message on the `AQAPP_NEWEMP` queue, which is what the `newemp.sql` script does.

Run the PL/SQL script to generate the event.

1. Log in to the database account `aqapp` where the `AQAPP_NEWEMP` queue was defined. (Refer to ["Task 2: Create the Application Queue AQAPP_NEWEMP"](#) on page B-42.) For example, assuming no connect string is necessary, specify the following command:

```
sqlplus aqapp/aqapp
```

2. Run the `newemp.sql` script:

```
SQL> START newemp.sql
```

The following message appears:

```
PL/SQL procedure successfully completed.
```

3. Exit SQL*Plus:

```
SQL> EXIT
```

B.7.5.4 Task 4: Verify Receipt of newEmployee Event

After some time (maybe several minutes depending on the system performance), a file appears in the `/tmp/fileapp_sub/write` directory, which represents the sending endpoint for the FTP adapter. The file is named after the pattern:

`app-name-timestamp`

1. Verify that the `newEmployee` event has been published and received by the `fileapp_sub` application. On UNIX, for example, perform the following commands:

```
$ cd /tmp/fileapp_sub/write
$ ls -l
total 2

-rw-rw-r-- 1 bstern svrtech 44 Dec 18 15:29 FILEAPP_SUB-1008718194783
```

The contents of the file can be displayed in different formats:

```
$ od -c FILEAPP_SUB-1008718194783
0000000 S c o t t T i g e r
0000020          001 001 | 0 5 / 0 1 / 2 0 0
0000040 1 | $ 5 2 3 0 8 . 7 5 $
```

or

```
$ od -x FILEAPP_SUB-1008718194783
0000000 5363 6f74 7420 5469 6765 7220 2020 2020
0000020 2020 2020 0101 7c30 352f 3031 2f32 3030
0000040 317c 2435 3233 3038 2e37 3524
```

2. Verify that this output corresponds to the D3L definition shown in "[Task 2: Create a D3L File for the FTP Adapter](#)" on page B-32 and the data enqueued by `newemp.sql`.
3. Repeat Step 2 on page B-47 to trigger and generate another event (file). The second time you trigger the event, the new file in the `/write` directory appears much faster (in approximately 3-4 seconds). This is because the adapter allocates and initializes all connections and data structures after processing the first message.
4. You have completed the use case.

B.7.6 Using Other Adapters in D3L and XML Modes

This section briefly describes how to use adapters other than the FTP adapter, and how to run them in XML mode instead of D3L mode. It contains these topics:

- [Using the HTTP, SMTP, or MQ Series Adapters in D3L Mode](#)

- [Using XML Mode](#)

B.7.6.1 Using the HTTP, SMTP, or MQ Series Adapters in D3L Mode

Perform the following steps to use the D3L use case with a different OracleAS Integration InterConnect adapter.

1. Enter another application name that indicates which adapter you are using in "[Task 7: Create the fileapp_sub Application](#)" on page B-40 (for example, smtpapp_sub.)
2. Specify the parameters needed for the particular adapter in Step 1 on page B-43. Refer to the installation documentation for the appropriate OracleAS Integration InterConnect Adapter.
3. In "[Task 4: Verify Receipt of newEmployee Event](#)" on page B-47, the verification process depends entirely on the adapter type, or more specifically, the exact sending endpoint defined.
4. Replace the fileapp_sub application name where ever it appears with the new application name.

The remaining steps are the same as deccribed in the previous section.

B.7.6.2 Using XML Mode

Perform the following steps to use XML as the operational mode of the OracleAS Integration InterConnect adapters.

1. Skip "[Task 2: Create a D3L File for the FTP Adapter](#)" on page B-32.
2. Define the following common view event attributes in Step 4 and Step 5 of "[Task 3: Create the newEmployee Event](#)" on page B-35:
 - a. Manually create a common data type (right-click + New) named NewEmpRec that has the same attributes as shown in the Create Event dialog on page B-35.
 - b. Import the common data typedata type defined in Step 2 instead of importing a D3L file.
3. Select XML instead of D3L in Step 3 of "[Select the Event to which to Subscribe](#)" on page B-40.
4. Select to import XML and choose the file newemp.dtd in Step 4 of "[Define the Application View](#)" on page B-40.
5. Perform Steps 2 through 4 in "[Define the Application View to Common View Mapping](#)" on page B-41 like you did Steps 2 through 7 in "[Define the Application View to Common View Mapping](#)" on page B-38.
6. Skip [Task 4: Copy the newemp.xml D3L File to the fileapp_sub Adapter Directory](#) on page B-44 and "[Task 5: Set the D3L file and Payload Type in the adapter.ini Adapter Initialization File](#)" on page B-44.

Note: Replacement step Step 2 assumes that you do not have the D3L file. However, as a shortcut, you can still define the common view event attributes as they were performed in Step 6 of "[Task 3: Create the newEmployee Event](#)" on page B-35.

B.8 Additional D3L Sample Files and DTD

This section contains these topics:

- [Additional D3L Sample Files](#)
- [D3L DTD](#)

B.8.1 Additional D3L Sample Files

This section provides several D3L sample files. These example files describe how to use the D3L language to define the content of native format messages.

- [Sample File with Structure VehicleRegistration](#)
- [Sample File with Structure Hierarchy PersonRecord](#)
- [Sample File with Structure ProductRecord](#)

B.8.1.1 Sample File with Structure VehicleRegistration

Sample file `msg-1.xml` represents a structure named `VehicleRegistration`. [Table B-4](#) describes the file fields and [Example B-22](#) shows `msg-1.xml` file contents.

Table B-4 *msg-1.xml File Fields*

Field	Description
SizeWeight	A fixed-length array of four signed, one-byte, little-endian integers, each aligned on two-byte boundaries (implying a one-byte padding between elements of the array.)
ProductCode	An unsigned, two-byte, big-endian integer aligned on two-byte boundaries.
VIN	An unsigned, eight-byte, big-endian integer aligned on two-byte boundaries.
PreviousOwners	A length-prefixed array of dates in the MMDDYYYY format (the length of the array is a signed, one-byte, little-endian integer with a two-byte alignment.)
Miles	An unsigned, two-byte, big-endian integer with a two-byte alignment.
DateProduced	A single date in the MMDDYYYY format.

Example B-22 *Sample File msg-1.xml with Structure EmployeeRegistration*

```
<?xml version="1.0" encoding="US-ASCII"?>
<message type="VehicleRegistration" name="Register" object="Vehicle">
  <date format="MMDDYYYY" id="Date_T">
    <padstring id="FixString10_T" length="10" padchar=' ' padstyle="none" />
  </date>
  <struct id="VehicleRegistration">
    <!-- Width x Length x Height x Weight (inch/lb) -->
    <field name="SizeWeight"><typeref type="ShortArray4_T" /></field>
    <field name="ProductCode"><unsigned2 align="2" endian="big" /></field>
    <field name="VIN"><unsigned8 align="2" endian="big" /></field>
    <field name="PreviousOwners"><typeref type="StringArray_T" /></field>
    <field name="Miles"><unsigned2 align="2" endian="big" /></field>
    <field name="DateProduced"><typeref type="Date_T" /></field>
  </struct>
  <fixarray id="ShortArray4_T" length="4">
    <unsigned2 align="2" endian="little" id="" />
  </fixarray>
  <unsigned1 align="2" endian="little" id="Short_T" />
  <pfxarray id="StringArray_T" length="Short_T">
    <typeref type="FixString10_T" />
  </pfxarray>
```

</message>

The following native format message examples show a hexadecimal and character representation of the same message, which can be parsed by the `msg-1.xml` D3L file:

- Hexadecimal format:

```
0000000 4500 b200 3400 8a0b 30d9 0000 0000 0072
0000020 55ff 0200 4a6f 6e65 732c 502e 2020 536d
0000040 6974 682c 522e 2020 5208 3131 2532 3225
0000060 3139 3939
```

- Character format:

```
0000000 E \0 262 \0 4 \0 212 013 0 331 \0 \0 \0 \0 \0 r
0000020 U 377 002 \0 J o n e s , P . S m
0000040 i t h , R . R \b 1 1 % 2 2 %
0000060 1 9 9 9
```

B.8.1.2 Sample File with Structure Hierarchy PersonRecord

Sample file `msg-2.xml` demonstrates a structure hierarchy named `PersonRecord`. [Table B-5](#) describes the file fields and [Example B-23](#) shows `msg-2.xml` file contents.

Table B-5 *msg-2.xml* File Fields

Field	Description
Name	A string delimited by a comma.
Age	An unsigned, one-byte integer.
DOB	A date in MMDDYYYY format, length prefixed by a signed, four-byte integer.
Phone	An unsigned, four-byte integer.
City	A structure named <code>CityRecord</code> that consists of the following fields: <ul style="list-style-type: none"> ■ Name ■ A string delimited by * ■ State ■ A string delimited by * ■ Country ■ A string delimited by * ■ Population ■ An unsigned, four-byte integer
State	A structure named <code>StateRecord</code> that consists of the following fields: <ul style="list-style-type: none"> ■ Name ■ A string delimited by a space ■ Capital ■ A string delimited by a space ■ Population ■ An unsigned, four-byte integer

Example B-23 *Sample File msg-2.xml with Structure PersonRecord*

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE message SYSTEM "d3l.dtd">
<message type="PersonRecord">
```

```

<signed4 id="s4" />
<struct id="CityRecord">
  <field name="Name"><limstring delimiter="*" /></field>
  <field name="State"><limstring delimiter="*" /></field>
  <field name="Country"><limstring delimiter="," /></field>
  <field name="Population"><unsigned4 /></field>
</struct>

<struct id="StateRecord">
  <field name="Name"><limstring delimiter=" " /></field>
  <field name="Capital"><limstring delimiter=" " /></field>
  <field name="Population"><unsigned4 /></field>
</struct>

<struct id="PersonRecord">
  <field name="Name"><limstring delimiter="," /></field>
  <field name="Age"><unsigned1 /></field>
  <field name="DOB">
    <date format="MMDDYYYY">
      <afxstring id="dobstr" length="s4" />
    </date>
  </field>
  <field name="Phone"><unsigned4 /></field>
  <field name="City"><typeref type="CityRecord" /></field>
  <field name="State"><typeref type="StateRecord" /></field>
</struct>

</message>

```

The following is a combined hexadecimal and character representation of a native message, which can be parsed by `msg-2.xml`:

```

000 2c4a 6f68 6e20 446f 652c 1e00 0000 000a ,John Doe,.....
020 3131 2f32 352f 3139 3635 0000 002c a155 11/25/1965...,U
040 2a50 6f72 746c 616e 642a 2a4f 522a 2c55 *Portland**OR*,U
060 5341 2c00 000f 4240 204f 7265 676f 6e20 SA,...B@_Oregon_
100 2053 616c 656d 2000 003d 0900 _Salem...=..

```

B.8.1.3 Sample File with Structure ProductRecord

Sample file `msg-3.xml` defines a structure named `ProductRecord`. [Table B-6](#) describes the file fields and [Example B-24](#) shows `msg-3.xml` file contents.

Table B-6 *msg-3.xml File Fields*

Field	Description
Manufacturer	A string delimited by a space.
Weight	A single-precision, floating-point number.
Widgets	A length-prefixed array of <code>WidgetRecord</code> structures. A <code>WidgetRecord</code> consists of: <ul style="list-style-type: none"> ▪ Name ▪ A string delimited by a space ▪ Color ▪ A string delimited by a space ▪ Weight ▪ A single-precision, floating point number

Example B-24 Sample File msg-3.xml with Structure ProductRecord

```

<?xml version="1.0" encoding="US-ASCII"?>

<!DOCTYPE message SYSTEM "d3l.dtd">

<message type="ProductRecord">
  <unsigned1 id="u1" />
  <unsigned2 id="u2" />
  <number id="pfxnum">
    <padstring length="8" padchar=" " padstyle="none" />
  </number>
  <pfxarray id="Unsigned1Tab" length="u1">
    <unsigned1 />
  </pfxarray>
  <pfxarray id="Signed4Tab" length="pfxnum">
    <unsigned4 />
  </pfxarray>
  <pfxarray id="StrTab" length="u1">
    <limstring delimiter=" " />
  </pfxarray>
  <struct id="WidgetRecord">
    <field name="Name"><limstring delimiter=" " /></field>
    <field name="Color"><limstring delimiter=" " /></field>
    <field name="Weight"><float /></field>
  </struct>
  <pfxarray id="WidgetTab" length="u2">
    <typeref type="WidgetRecord" />
  </pfxarray>
  <struct id="ProductRecord">
    <field name="Manufacturer"><limstring delimiter=" " /></field>
    <field name="Weight"><float /></field>
    <field name="Widgets"><typeref type="WidgetTab" /></field>
  </struct>

</message>

```

B.8.2 D3L DTD

[Example B-25](#) shows the DTD to which D3L (XML) files must conform.

Example B-25 D3L DTD

```

<!ENTITY % Name      "CDATA"      >
  <!ENTITY % Number  "NMTOKEN"    >
  <!ENTITY % Comment "CDATA"      >
  <!ENTITY % DelimiterChar
    "CDATA"
  >
  <!ENTITY % QuotationCharAttribute
    "
    quote %DelimiterChar; #IMPLIED
    "
  >
  <!-- ===== -->
  <!ENTITY % GenericAttributes
    "
    name %Name;          #IMPLIED
    comment %Comment;   #IMPLIED
    id ID                #IMPLIED
  >

```



```

"
>
<!ENTITY % FieldAttributes
"
    name    %Name;          #REQUIRED
    comment %Comment;      #IMPLIED
    id      ID              #IMPLIED
"
>
<!ENTITY % NonTypeAttributes
"
    name    %Name;          #IMPLIED
    comment %Comment;      #IMPLIED
"
>
<!-- ===== -->
<!ENTITY % StructAttributes
"
    %GenericAttributes;
    %QuotationCharAttribute;
"
>
<!-- ===== -->
<!ENTITY % Align
    "%Number;"
>
<!ENTITY % IntegerSize
    "( 1 | 2 | 4 | 8 )"
>
<!ENTITY % Endian
    "( big | little )"
>
<!ENTITY % IntegerAttributes
"
    %GenericAttributes;
    endian %Endian;        'big'
"
>
<!ENTITY % IntegerTypes
    " signed1 | unsigned1
    | signed2 | unsigned2
    | signed4 | unsigned4
    | signed8 | unsigned8
"
>
<!ENTITY % FloatAttributes
"
    %GenericAttributes;
"
>
<!ENTITY % FloatTypes
    " float | double
"
>
<!-- ===== -->
<!ENTITY % PadStyle
    "( head | tail | none )"
>
<!ENTITY % PadChar
    "CDATA"

```

```

>
<!ENTITY % StringAttributes
"
    %GenericAttributes;
"
>
<!ENTITY % SimpleStringAttributes
"
    %StringAttributes;
"
>
<!ENTITY % TerminatedStringAttributes
"
    %StringAttributes;
    endchar    %DelimiterChar; #REQUIRED
"
>
<!ENTITY % QuotedTerminatedStringAttributes
"
    %StringAttributes;
    %QuotationCharAttribute;
    endchar    %DelimiterChar; #REQUIRED
"
>
<!ENTITY % PaddedStringAttributes
"
    %StringAttributes;
    length     %Number;          #REQUIRED
    padchar    %PadChar;         #REQUIRED
    padstyle   %PadStyle;        #REQUIRED
"
>
<!ENTITY % PrefixedStringAttributes
"
    %StringAttributes;
    length     IDREF             #REQUIRED
"
>
<!ENTITY % DelimitedStringAttributes
"
    %StringAttributes;
    delimiter  %DelimiterChar; #REQUIRED
"
>
<!ENTITY % StringTypes
    "padstring | pfxstring | limstring | qtdtermstring | termstring |
simplestring "
>
<!-- ===== -->
<!ENTITY % DateFormat
    "( DDMYY | DDMYYYY | MMDDY | MMDDYYYY )"
>
<!ENTITY % DateAttributes
"
    %GenericAttributes;
    format    %DateFormat;      #REQUIRED
"
>
<!-- ===== -->
<!ENTITY % NumberAttributes

```

```

"
    %GenericAttributes;
"
>
<!-- ===== -->
<!ENTITY % ArrayAttributes
"
    %GenericAttributes;
"
>
<!ENTITY % FixedArrayAttributes
"
    %ArrayAttributes;
    length      %Number;          #REQUIRED
"
>
<!ENTITY % PrefixedArrayAttributes
"
    %ArrayAttributes;
    length      IDREF             #REQUIRED
"
>
<!ENTITY % DelimitedArrayAttributes
"
    %ArrayAttributes;
    contchar    %DelimiterChar; #REQUIRED
    endchar     %DelimiterChar; #REQUIRED
"
>
<!ENTITY % ImplicitArrayAttributes
"
    %ArrayAttributes;
"
>
<!-- ===== -->
<!ENTITY % ScalarElements
"
    signed1 | unsigned1
    | signed2 | unsigned2
    | signed4 | unsigned4
    | signed8 | unsigned8
    | float | double
    | date | number
    | padstring
    | pfxstring
    | limstring
    | termstring
    | qtdtermstring
    | simplestring
"
>
<!ENTITY % TypeElements
"%ScalarElements;
| struct
| fixarray
| pfxarray
| limarray
| imparray
"
>
<!-- ===== -->

```

```

<!ENTITY % FieldElements
    "%TypeElements;"
>
<!ENTITY % MessageElements
    "%TypeElements;"
>
<!ENTITY % StructElements
    "field | pad"
>
<!ENTITY % ArrayElements
    "%ScalarElements; | struct"
>
<!ENTITY % ImplicitArrayElements
    "%ArrayElements; | limarray"
>
<!-- ===== -->
<!ELEMENT message ( %MessageElements; )* >
    <!ATTLIST message
        name      %Name;          #REQUIRED
        object    CDATA           #REQUIRED
        type      IDREF           #REQUIRED
        comment   %Comment;       #IMPLIED
        id        ID              #IMPLIED
        header    CDATA           #IMPLIED
        value     CDATA           #IMPLIED
        magic     CDATA           #IMPLIED
        startsat  %Number;        #IMPLIED
        reply     (Y|N)           "N"
        %QuotationCharAttribute;
    >
<!-- ===== -->
<!ELEMENT struct ( %StructElements; )* >
    <!ATTLIST struct
        %StructAttributes;
    >
<!-- ===== -->
<!ELEMENT field ( typeref | %FieldElements; ) >
    <!ATTLIST field
        %FieldAttributes;
    >
<!-- ===== -->
<!ELEMENT signed1 EMPTY >
    <!ATTLIST signed1
        %IntegerAttributes;
        size  %IntegerSize;  #FIXED  "1"
        align %Align;        "1"
    >
<!ELEMENT unsigned1 EMPTY >
    <!ATTLIST unsigned1
        %IntegerAttributes;
        size  %IntegerSize;  #FIXED  "1"
        align %Align;        "1"
    >
<!ELEMENT signed2 EMPTY >
    <!ATTLIST signed2
        %IntegerAttributes;
        size  %IntegerSize;  #FIXED  "2"
        align %Align;        "2"
    >
<!ELEMENT unsigned2 EMPTY >

```

```

        <!ATTLIST unsigned2
            %IntegerAttributes;
            size %IntegerSize; #FIXED "2"
            align %Align;      "2"
        >
<!ELEMENT signed4 EMPTY >
    <!ATTLIST signed4
        %IntegerAttributes;
        size %IntegerSize; #FIXED "4"
        align %Align;      "4"
    >
<!ELEMENT unsigned4 EMPTY >
    <!ATTLIST unsigned4
        %IntegerAttributes;
        size %IntegerSize; #FIXED "4"
        align %Align;      "4"
    >
<!ELEMENT signed8 EMPTY >
    <!ATTLIST signed8
        %IntegerAttributes;
        size %IntegerSize; #FIXED "8"
        align %Align;      "8"
    >
<!ELEMENT unsigned8 EMPTY >
    <!ATTLIST unsigned8
        %IntegerAttributes;
        size %IntegerSize; #FIXED "8"
        align %Align;      "8"
    >
<!-- ===== -->
<!ELEMENT float EMPTY >
    <!ATTLIST float
        %FloatAttributes;
        align %Align;      "4"
    >
<!ELEMENT double EMPTY >
    <!ATTLIST double
        %FloatAttributes;
        align %Align;      "8"
    >
<!-- ===== -->
<!ELEMENT simplestring EMPTY >
    <!ATTLIST simplestring
        %SimpleStringAttributes;
    >
<!ELEMENT qtdtermstring EMPTY >
    <!ATTLIST qtdtermstring
        %QuotedTerminatedStringAttributes;
    >
<!ELEMENT termstring EMPTY >
    <!ATTLIST termstring
        %TerminatedStringAttributes;
    >
<!ELEMENT padstring EMPTY >
    <!ATTLIST padstring
        %PaddedStringAttributes;
    >
<!ELEMENT pfxstring EMPTY >
    <!ATTLIST pfxstring
        %PrefixedStringAttributes;

```

```
>
<!ELEMENT limstring EMPTY >
  <!ATTLIST limstring
    %DelimitedStringAttributes;
  >
<!-- ===== -->
<!ELEMENT fixarray ( typeref | %ArrayElements; ) >
  <!ATTLIST fixarray
    %FixedArrayAttributes;
  >
<!ELEMENT pfixarray ( typeref | %ArrayElements; ) >
  <!ATTLIST pfixarray
    %PrefixedArrayAttributes;
  >
<!ELEMENT limarray ( typeref | %ArrayElements; ) >
  <!ATTLIST limarray
    %DelimitedArrayAttributes;
  >
<!ELEMENT imparray ( typeref | %ImplicitArrayElements; ) >
  <!ATTLIST imparray
    %ImplicitArrayAttributes;
  >
<!-- ===== -->
<!ELEMENT date ( typeref | %StringTypes; ) >
  <!ATTLIST date
    %DateAttributes;
  >
<!-- ===== -->
<!ELEMENT number ( typeref | %StringTypes; ) >
  <!ATTLIST number
    %NumberAttributes;
  >
<!-- ===== -->
<!ELEMENT typeref EMPTY >
  <!ATTLIST typeref
    %NonTypeAttributes;
    type IDREF #REQUIRED
  >
<!-- ===== -->
<!ELEMENT pad EMPTY >
  <!ATTLIST pad
    %NonTypeAttributes;
    length %Number; #REQUIRED
  >
<!-- ===== -->
```

Transformations

C.1 OracleAS Integration InterConnect Transformations

This appendix provides a table with the OracleAS Integration InterConnect transformations.

C.1.1 Copy Fields

Copy the source fields into the destination fields.

Parameters:

None

C.1.2 Copy Object

Copy the source object into the destination object.

Parameters:

None

C.1.3 Concat Fields

Concatenate the source fields, and copy into the destination fields.

Parameter	Description
<code>prefix</code>	An optional prefix to the concatenated string.
<code>separator</code>	The separator, a string of characters, that separate source fields in the concatenated string.
<code>suffix</code>	An optional suffix to the concatenated string.

C.1.4 Expand Fields

Expand the source field into the destination fields.

Parameter	Description
<code>delimiter</code>	The delimiter or string of characters around which the source field should be separated.

C.1.5 Set Constant

Copy a constant into the destination fields.

Parameter	Description
<code>constant</code>	The constant to be copied.

C.1.6 True Conditional Lookup XRef

Find the source field in a cross-reference table. If the condition is satisfied, then copy it into the destination field.

Parameter	Description
<code>condition</code>	The condition for this parameter.
<code>table</code>	The cross-reference table.
<code>pass through</code>	If there is no corresponding cross-reference, and the parameter is <code>true</code> , then the destination field is set to the source field. If this parameter is <code>false</code> , then the destination field is set to null.

C.1.7 True Conditional Lookup DVM

Find the source field in a domain value map table. If the condition is satisfied, then copy it into the destination field.

Parameter	Description
<code>condition</code>	The condition for this parameter.
<code>table</code>	The domain value map table.
<code>pass through</code>	If there is no corresponding domain value map and this parameter is set to <code>true</code> , then the destination field is set to the source field. If this parameter is set to <code>false</code> , then the destinations field is set to null.

C.1.8 Conditional Copy

Copy the source fields into the destination fields if the expression is satisfied.

Parameter	Description
<code>expression</code>	The expression.
<code>only copy on true</code>	If this parameter is set to <code>true</code> and the expression evaluates to <code>false</code> , then nothing is copied. If this parameter is set to <code>false</code> and the expression evaluates to <code>false</code> , then the second input object is copied.

C.1.9 True Conditional Copy

Copy the source fields into the destination fields if the condition is satisfied.

Parameter	Description
condition	The condition for this parameter.

C.1.10 True Conditional Concat

Concatenate the source fields into the destination field if the condition is satisfied.

Parameter	Description
condition	The condition for this parameter.
prefix	An optional prefix to the concatenated string.
separator	The separator, a string of characters, that separate source fields in the concatenated string.
suffix	An optional suffix to the concatenated string.

C.1.11 True Conditional To Number

Convert the sign, value, and precision source fields into a number, and copy it into the destination field if the condition is satisfied.

Parameter	Description
condition	The condition for this parameter.
int length	The number of digits before the decimal point excluding the sign.
dec length	The number of digits after the decimal point.
character	The padding character.
DVM	An optional domain value map to lookup decimal point character.

C.1.12 False Conditional Copy

Copy the source fields into the destination fields if condition is not satisfied.

Parameter	Description
condition	The condition for this parameter.
condition	The condition for this parameter.
condition	The condition for this parameter.

C.1.13 False Conditional Concat

Concatenate the source fields into the destination field if the condition is not satisfied.

Parameter	Description
<code>condition</code>	The condition for this parameter.
<code>prefix</code>	An optional prefix to the concatenated string.
<code>separator</code>	The separator, a string of characters, that separate source fields in the concatenated string.

C.1.14 False Conditional To Number

Convert the sign, value, and precision source fields into a number and copy it into the destination field if condition is not satisfied.

Parameter	Description
<code>condition</code>	The condition for this parameter.
<code>int length</code>	The number of digits before the decimal point excluding the sign.
<code>dec length</code>	The number of digits after the decimal point.

C.1.15 To Number

Convert the sign, value, and precision source fields into a number, and copy it into the destination field.

Parameter	Description
<code>int length</code>	The number of digits before the decimal point excluding the sign.
<code>dec length</code>	The number of digits after the decimal point.
<code>character</code>	The padding character.

C.1.16 Substring

Copy a substring of the source field into the destination field.

Parameter	Description
<code>begin index</code>	The index at which the substring begins.
<code>length</code>	An optional length of the substring.
<code>begin index</code>	The index at which the substring begins.

C.1.17 Char Replace

Replace characters in the source field, and copy it into the destination field.

Parameter	Description
<code>targets</code>	The string of characters to replace.
<code>replacements</code>	The string of replacement characters.

Parameter	Description
targets	The string of characters to replace.

C.1.18 String Replace

Replace each occurrence of a string in the source field, and copy the replacement into the destination field.

Parameter	Description
targets	The string of characters to replace.
replacements	The string of replacement characters.
targets	The string of characters to replace.

C.1.19 LTrim

Delete source field characters starting from the left until a character from the set is found, and copy the remaining string into the destination field.

Parameter	Description
characters	The string of characters to seek that stop the deletion.
characters	The string of characters to seek that stop the deletion.
characters	The string of characters to seek that stop the deletion.

C.1.20 RTrim

Delete source field characters starting from the right until a character from the set is found, and copy the remaining string into the destination field.

Parameter	Description
characters	The string of characters to seek that stop the deletion.
characters	The string of characters to seek that stop the deletion.
characters	The string of characters to seek that stop the deletion.

C.1.21 LPad

Pad source field starting from the left for a given length, and copy it into the destination field.

Parameter	Description
length	The padding length.
character	An optional character to pad with, default is <space>.
length	The padding length.

C.1.22 RPad

Pad source field starting from the right for a given length, and copy it into the destination field.

Parameter	Description
length	The padding length.
character	An optional character to pad with, default is <space>.
length	The padding length.

C.1.23 Lookup XRef

Lookup the source field in a cross-reference table, and copy it into the destination field.

Parameter	Description
table	The cross-reference table.
pass through	If there is no corresponding cross-reference and this parameter is set to <code>true</code> , then the destination field is set to the source field. If this parameter is set to <code>false</code> , then the destination field is set to <code>null</code> .
table	The cross-reference table.

C.1.24 Delete XRef

Delete the source field from a cross-reference table.

Parameter	Description
table	The cross-reference table.
table	The cross-reference table.
table	The cross-reference table.

C.1.25 Lookup DVM

Look up the source field in a domain value map table, and copy it into the destination field.

Parameter	Description
table	The domain value map table.
pass through	If there is no corresponding domain value map and this parameter is set to <code>true</code> , then the destination field is set to the source field. If this parameter is set to <code>false</code> , then the destination field is set to <code>null</code> .
table	The domain value map table.

C.1.26 Truncate

Truncate source field starting from the right for a given length, and copy it into the destination field.

Parameter	Description
length	The length to truncate.
length	The length to truncate.
length	The length to truncate.

C.1.27 Increment

Increment a counter, and copy the incremented value into the destination field.

Parameter	Description
start value	The initial counter value.
counter	The name of the counter. The name should distinguish it from other counters that may be at different values at a given time and may have a different step size.
step size	The increment size.

C.1.28 DatabaseOperation

Apply SQL or PL/SQL operations to the source fields, and copy the result to the destination fields. This transformation can be processed on any database, including the hub or database adapter instance. Connect to the database given by the connectivity parameters and bind the input variables to the corresponding bind variables of the SQL or PL/SQL given by the `operation` parameters.

The statement is then run. Upon successful execution, the results are copied to the destination field of the transformation. The connection to the database is then closed and the result of the transformation is returned.

Parameter	Description
db user	The database user name.
db password	The password of the database user.
db host	The hostname of the database.

Note: For the PL/SQL type of transformation, the following syntax of the PL/SQL statement is assumed:

- IN parameters are specified with a ?I
- OUT parameters are specified with a ?O:<T>
- IN/OUT parameters are specified with a ?IO:<T>

<T> is a single character type specifier denoting the type of the variable. Valid variable values are:

- S: String
 - I: Integer
 - F: Float
 - D: Double
 - T: Date
 - B: Binary
-
-

Troubleshooting OracleAS Integration InterConnect

This appendix describes common problems that you might encounter when using OracleAS Integration InterConnect and explains how to solve them. It also gives detailed instructions on how to diagnose OracleAS Integration InterConnect problems. It contains the following topics:

- [Problems and Solutions](#)
- [Need More Help?](#)

D.1 Problems and Solutions

This section describes common problems and solutions. It contains the following topics:

- [iStudio Fails to Connect](#)
- [Mappings in iStudio](#)
- [Metadata Not Editable in iStudio](#)
- [Subscribing Adapter Does Not Receive Messages From the Hub](#)
- [Messages Are Not Getting Delivered to a Spoke Application](#)
- [OracleAS Integration InterConnect Repository Does Not Start](#)
- [DB Adapter Does Not Pick Up Messages That Have Been Published From the Database](#)
- [FTP Adapter Cannot Match Incoming Message With Any D3L Definition](#)
- [AQ Adapter Does Not Pick Up Message From the Spoke AQ](#)
- [CBR Issue with DB Adapter](#)

D.1.1 iStudio Fails to Connect

iStudio is not connecting to the repository. For example, a message pops up saying "Failed to connect to the repository".

Problem 1

Incorrect hub database parameters entered in the Hub Information dialog.

Solution 1

iStudio uses the user name, password, and connect string entered in the 'Hub Information' dialog to establish a connection with the repository. Ensure that you have provided the correct hub database parameters, `<hostname>:<tns_listener_port>:<db_sid>`, for connecting to the Infrastructure Database. For example, `myhost.us.mycompany.com:1521:orcl`.

See Also: Section 2.5.1, "Creating a New Project" for more details on the hub database parameters

Problem 2

Repository is down.

Solution 2

Ensure that the repository is properly started and check the `$ORACLE_HOME/integration/<version>/repository/reposlog.txt` file for any problems.

Problem 3

Repository is running behind a firewall.

Solution 3

Ensure that you configure the repository to use a specific RMI communication port. This is configurable in the `$ORACLE_HOME/integration/<version>/repository/repository.ini` file.

D.1.2 Mappings in iStudio

I have changed the mapping, but the adapter(s) still seem to use the old information.

Problem 1

Metadata is not refreshed.

Solution 1

In the File menu, select **Sync Adapters**. This pushes the updated metadata to the adapters.

Solution 2

Alternatively, you could perform the following steps to change the behavior after transformation:

1. Stop the adapters.
2. In the `adapter.ini` file, set the value of the parameter `agent_delete_file_cache_at_startup` to **True**, to delete all cached metadata at startup.

This parameter specifies whether to delete the cached metadata during startup. If any of the agent caching methods such as metadata caching, DVM table caching, or lookup table caching is enabled, then metadata from the repository is cached locally on the file system. However, if you change some metadata or DVM table using iStudio and you want the Adapter to use those changes the next time it is started, then you can either delete the cache files or set this parameter to true before restarting.

3. Restart the adapters.

D.1.3 Metadata Not Editable in iStudio

The metadata is not editable in iStudio as the "Edit" option is greyed out.

Problem 1

The metadata in the repository might belong to a different metadata owner than the one specified in the `$ORACLE_HOME/integration/<version>/repository/repository.ini` file.

Solution 1

Ensure that the "owner" of the data is same in both repositories. Data types have an owner, usually, that is "OAI". But you can change it in the `repository.ini` file. Only data that is owned by the local repository owner may be edited.

The metadata owner information is stored along with the objects at the time of their creation. If the repository owner name does not match the creation time owner name, then you will notice that the edit buttons are greyed out. You will need to edit the `repository.ini` file to change the owner name to the one used during creation and restart the repository. You will also need to reestablish the iStudio connection to the repository.

D.1.4 Subscribing Adapter Does Not Receive Messages From the Hub

The subscribing adapter is not able to pickup any message from the hub queue (`oai_hub_queue`). The messages are queued up in the hub queue but aren't subscribed by the subscribing adapter.

Problem 1

The value for the `application`, `agent_subscriber_name`, and `agent_message_selector` parameter in the `$ORACLE_HOME/integration/<version>/adapters/<adapter_name>/adapter.ini` file doesn't match the application name in iStudio.

Solution 1

Ensure that the value for the `application`, `agent_subscriber_name`, and `agent_message_selector` parameter in the `$ORACLE_HOME/integration/<version>/adapters/<adapter_name>/adapter.ini` file match the application name in iStudio.

Problem 2

Hub Queue name of the application may be longer than 20 characters.

Solution 2

Apply patch 2659228.

D.1.5 Messages Are Not Getting Delivered to a Spoke Application

If an adapter successfully received a message from the hub, but encounters a problem while transforming the message or delivering the message to the spoke application, then it will try to reprocess/resent the message in a certain interval until the problem has been resolved and the message has been successfully delivered. In most cases, this unfortunately means that all other pending messages that need to be processed to the spoke application won't get delivered until this one message has cleared out of the system.

Problem

OracleAS Integration InterConnect guarantees in-order-delivery of messages, meaning that if an adapter encounters a problem while processing a message, it will try to re-send the message until the problem has been resolved and the message can be delivered successfully. All other pending messages can't get delivered as this would change the order of delivery.

Solution 1

Refer to the error message and the `oailog.txt` file in `$ORACLE_HOME/integration/<version>/adapters/<adapter_name>/logs/` directory, to resolve the error message.

Solution 2

Use the `$ORACLE_HOME/integration/<version>/adapters/<adapter_name>/ignoreErrors` script to drop the problem message out of the system. The `ignoreErrors` script will reconfigure the adapter, so that it ignores a specific error code. The adapter will then drop all messages that fail with that specific error code rather than trying to resend them, which ensures that other pending messages can be processed.

D.1.6 OracleAS Integration InterConnect Repository Does Not Start

The OracleAS Integration InterConnect repository does not start properly and there are java exceptions in the `$ORACLE_HOME/integration/<version>/repository/reposlog.txt` file. For example, "Could not register with the hub database. Please check your hub database parameters." Error message: "Retrying in 10 sec.java.sql.SQLException: Io exception: The Network Adapter could not establish the connection".

Problem 1

The information in the `$ORACLE_HOME/integration/<version>/hub/hub.ini` file is not valid.

Solution 1

Provide the correct information in the `$ORACLE_HOME/integration/<version>/hub/hub.ini` file.

```
[Database]
```

```
hub_username=ichub
```

```
encrypted_hub_password=<encrypted_password> use $ORACLE_HOME/integration/<version>/bin/encrypt for encryption
```

```
hub_host=<hub_host_name>
```

```
hub_instance=<database_sid>
```

```
hub_port=<tns_listener_port>
```

```
hub_use_thin_jdbc=true
```

```
[Repository Info]
```

```
repository_name=InterConnectRepository1012
```

Problem 2

The hub schema does not exist on the hub database.

Solution 2

Run the `$ORACLE_HOME/integration/<version>/repository/hubschema` script to create the hub schema.

D.1.7 DB Adapter Does Not Pick Up Messages That Have Been Published From the Database

The database adapter does not pick up messages that have been published from the database, even though the publishing PL/SQL procedure was performed successfully and the message object record exists in the `MESSAGEOBJECT` table.

Problem

You might have published the message for the wrong application. The value provided for `srcAppName+partition id` parameter in the publishing PL/SQL procedure `pub_<event/procedure_name>_<metadata_owner>_<version>` does not match the application parameter value in the `$ORACLE_HOME/integration/<version>/adapters/<adapter_name>/adapter.ini` file.

Solution

Ensure that the value provided for the `srcAppName+partition id` parameter matches the value of the application parameter in the `$ORACLE_HOME/integration/<version>/adapters/<adapter_name>/adapter.ini` file.

D.1.8 FTP Adapter Cannot Match Incoming Message With Any D3L Definiton

The FTP adapter throws an error message that it cannot match the incoming message with any of the D3L defintions, and drops the message.

Problem 1

The `ota.d3ls` parameter is either empty or points to the wrong D3L file(s).

Solution 1

Provide a valid D3L file for `ota.d3ls` (or list of D3L files).

Problem 2

The D3L header does not contain the correct information for OracleAS Integration InterConnect event mapping.

Solution 2

Ensure that the

- name tag in the D3L header matched the BusinessObject event name in iStudio.
- Object Name field in the Define application View window of the Publish/Subscribe/Invoke/Implement wizard is not empty.
- object tag matches the value of the Object Name field in the Define application View window of the Publish/Subscribe/Invoke/Implement wizard.

D.1.9 AQ Adapter Does Not Pick Up Mesage From the Spoke AQ

The AQ adapter does not pick up message that have been enqueued to a spoke AQ.

Problem 1

You have not provided the name of the spoke queue in the Deploy Tab in iStudio for this integration point.

Solution 1

Provide the name of the spoke queue. The AQ adapter log file should contain a line "<timestamp>: AQ Adapter: created a reader for queue <queue_name>".

Problem 2

The spoke queue might be a multiconsumer queue (JMS Topic) and you might connect to it with an incorrect consumer name.

Solution 2

Ensure that the `aq_bridge_consumer_name` parameter in the `$ORACLE_HOME/integration/<version>/adapters/<adapter_name>/adapter.ini` file is correct and matches the name of the consumer that the message was enqueued for.

D.1.10 CBR Issue with DB Adapter

I created an event with the name `ProdRelease`. The root element name is `BatchProdRelease`. When I restart the adapter, it displays the following error:

```
"An exception occurred while evaluating the CBR expression: (ProdRelease_CO:OAI_META/V1.BatchProdRelease_CO:OAI_META/V1.Header.PlantCode == "GFL").  
Following is a detailed message. oracle.oai.agent.common.AgentRuntimeException:  
Specified attribute (BatchProdRelease_CO:OAI_META/V1) does not exist in Data  
Type (ProdRelease_CO:OAI_META/V1)."
```

Problem

The root element name is a substring of the event name.

Solution

Ensure that the root element name does not have a substring of the event name. The root element name in this situation is `BatchProdRelease` and the event name is `ProdRelease`. This causes the adapter to fail. A workaround for this is to change either the root element name or event name.

D.2 Need More Help?

You can find more solutions on Oracle *MetaLink*, <http://metalink.oracle.com>. If you do not find a solution for your problem, log a service request.

See Also:

- *Oracle Application Server Release Notes*, available on the Oracle Technology Network:
<http://www.oracle.com/technology/documentation/index.html>

Glossary

Advanced Business Application Programming

A programming language developed by SAP for application development purposes.

adapter

Enables third-party environments to participate in application integration. An adapter has two major tasks:

- Provide connectivity between an application and the hub.
- Transform and route messages between the application and the hub.

adapter.ini file

An initialization parameter file that an adapter uses at startup to connect to an application.

advanced queuing adapter

Enables an Oracle Advanced Queuing application to be integrated with other applications using OracleAS Integration InterConnect.

agent

A subcomponent of an adapter that handles runtime instructions. The agent is independent of the application to which the adapter connects. The agent focuses on the integration scenario based on the integration metadata in the repository.

application

A component integrated with OracleAS Integration InterConnect.

application view

A native view translated into the syntax used by an adapter. . Each application has its own application view of data that allows it to participate in the integration. The application view of data uses transformations to map into the common view.

bridge

A subcomponent of an agent adapter that transfers data between the application and OracleAS Integration InterConnect. The bridge is the protocol/application-specific piece of the adapter that communicates with the application.

Business Application Programming Interface

Standardized programming interface that enables external applications to access the business processes and data of the R/3 system.

business object

A collection of logically related integration points.

cipher suites

A set of cryptographic algorithms. SSL supports different cryptographic algorithms, or ciphers, for tasks such as authenticating the server and client to each other, transmitting certificates, and establishing session keys. Clients and servers support different cipher suites depending on factors such as the SSL version supported, company policies regarding permissible encryption strength, and government restrictions on export of SSL-enabled software.

common view

A view that is syntactically and semantically in the OracleAS Integration InterConnect format. The common view:

- Identifies the list of integration points for applications. Applications participate in integration by binding to one or more common view integration points, such as creating a purchase order and creating a new customer.
- Eliminates the complexity of multiple integration points between applications.

content-based routing

Messages routed to specific applications based on business rules or message content.

cross-reference tables

Keys for corresponding entities created in different applications can be correlated through cross-referencing.

D3L

Data Definition Description Language. An XML-based message description language that describes application message information in its native format, also known as its native view.

database adapter

Enables an Oracle Application, typically PL/SQL-based, to be integrated with other applications using OracleAS Integration InterConnect.

design time

During the design phase, a business analyst uses iStudio to define the integration objects, applications that participate in the integration, and the specifications of the data exchanged between applications.

domain value maps

Code tables mapped across different systems.

DTD

Document Type Definition. A set of rules that defines the allowable structure of an XML document. DTDs are text files that derive their format from SGML and are either embedded within an XML document.

EAI

Enterprise Application Integration. The integration of applications and business processes within the same company.

endpoints

The physical destination points for messages exchanged between OracleAS Integration InterConnect and an application.

event

An integration point used to pattern the publish/subscribe model. An event has associated data that is the common view of all the data to be exchanged through this event. An event can be published or subscribed by an application.

event map

Allows application data to be mapped to an OracleAS Integration InterConnect event without the application having to know about the OracleAS Integration InterConnect event itself.

FTP adapter

Enables an FTP to be integrated with other applications using OracleAS Integration InterConnect.

HTTP

Hypertext Protocol Transfer. The underlying format, or protocol, used by the Web to format and transmit messages and determine what actions Web servers and browsers should take in response to various commands. HTTP is the protocol used between Oracle Application Server and clients.

HTTP adapter

Enables an Oracle HTTP application to be integrated with other applications using OracleAS Integration InterConnect. This adapter is useful in all EAI environments that use the HTTP transport protocol.

IDoc Type

Indicates the SAP format used to transfer the data for a business transaction. An IDoc is a real business process in the form of an IDoc type. An IDoc type is described using the following components:

- A control record is the format of the control record, which is identical for all IDoc types.
- One or more data records consist of a fixed administration part and a data part (segment). The number and format of the segments can be different for each IDoc type.
- Status records describe the processing stages which an IDoc can pass through and have identical formats for each IDoc type.

invoke/implement model

An application invokes a procedure by sending data out to the OracleAS Integration InterConnect hub and expects return of the result from an application implementing the procedure. An application implements a procedure by receiving data from the OracleAS Integration InterConnect hub and returns the result once the procedure has been performed. In iStudio, a procedure is used to model this scenario.

IMAP4

Internet Message Access Protocol 4. IMAP4 is a standard protocol for accessing e-mail from a local server. IMAP4 is a client/server protocol in which e-mail is received and held for users by their Internet server. Users can view just the heading and sender of

the e-mail, and then decide whether to download the e-mail. Users can also create and manipulate folders or mailboxes on the server, delete messages, or search for certain parts or an entire note. IMAP requires continual access to the server during the time that users work with their e-mail.

iStudio

A design time integration specification tool targeted at business analysts. This tool helps business analysts specify the integration logic at a functional level, instead of a technical coding level. iStudio exposes the integration methodology using simple wizards and reduces, or eliminates, the need for writing code to specify the integration logic. This reduces the total time required to complete an integration.

metadata

A definition or description of data (essentially, data about data).

MQ Series adapter

Enables OracleAS Integration InterConnect to send message to and receive messages from the MQ Series queues and topics.

native view

An application's message information in its native format (for example, SAP IDoc). Native events are both syntactically and semantically in the native format of the application, and are defined external to OracleAS Integration InterConnect.

Oracle Wallet Manager

A Java-based application that security administrators use to manage public-key security credentials on clients and server. Security credentials consist of a public/private key pair, a certificate, and a trustpoint.

Oracle Workflow

Integrated with OracleAS Integration InterConnect and is used for business process collaborations across two or more applications.

OracleAS Integration InterConnect

The integration hub that coordinates the communication and transformation of messages between two or more heterogeneous applications. OracleAS Integration InterConnect defines business events, their associated data, and any transformations required to map one application's view of a business object to another's view.

payload

The data sent between applications. For example, the payload data for a purchase order sent from one application to another application may include the product name, the quantity ordered, and the price.

persistence

The ability to save data and restore it when needed.

procedure

An integration point used to pattern the invoke/implement model. A procedure has associated data that is the common view of all the data to be exchanged through this procedure. A procedure can be invoked or implemented by an application.

project

Encapsulates all the integration logic for one integration scenario.

proxy host

A server through which messages sent to remote Web servers must pass. A proxy server also prevents users outside a company's firewall from breaking into an organization's private network.

publish/subscribe model

An application publishes an event when it sends data out to the OracleAS Integration InterConnect hub without knowing the destination applications. Furthermore, data is not expected in return. An application subscribes to an event if it receives the data from the OracleAS Integration InterConnect hub regardless of who sent the data. Furthermore, it does not send out any data in return. Events in iStudio are used to model this scenario.

realm

Realms enable the protected resources on a server to be partitioned into a set of protection spaces, each with its own authentication scheme and authorization database.

repository

The repository has the following functionality:

- At design time, all integration logic defined in iStudio is stored in tables in the repository metadata.
- At runtime, the repository provides access to this metadata for an metadata to integrate applications.

The repository server is deployed as a standalone Java application running outside the database. The repository schema is a set of tables in the Oracle Application Server Infrastructure.

RMI

Remote Method Invocation. An interaction scheme for distributed objects written in Java. It enables a Java program running on one computer to access the methods of another Java program running on another computer.

runtime

For each application participating in a specific integration, OracleAS Integration InterConnect attaches one or more adapters to it. At runtime, the adapters retrieve the metadata from the repository to determine the format of messages, perform transformations between the various data formats, and route the messages to the appropriate queues in the OracleAS Integration InterConnect hub.

SMTP

Simple Mail Transfer Protocol. A TCP/IP protocol for sending and receiving e-mail. SMTP is typically used with one of two other protocols, Post Office Protocol 3 (POP3) or Internet Message Access Protocol (IMAP), that enable users to save messages in a server mailbox and periodically download them. Users typically use a program that uses SMTP for sending e-mail and either POP3 or IMAP for receiving messages on their local server.

SMTP adapter

The SMTP adapter enables an SMTP application to be integrated with other applications using OracleAS Integration InterConnect. This adapter is useful in all EAL environments where e-mail used the IMAP4 and SMTP transport protocols.

SSL

Secure Sockets Layer. SSL is a standard for the secure transmission of documents over the Internet using HTTPS (secure HTTP). SSL uses digital signatures to ensure that transmitted data is not tampered with.

tracking fields

One or more application view fields in the context of a particular event. Used to track the event instances at runtime.

wallet

A wallet is an abstraction used to store and manager security credentials for an individual entity. It implements the storage and retrieval of credentials for use with various cryptographic services.

workspace

Stores user settings and preferences such as application login credentials and last opened project.

XML

eXtensible Markup Language. XML is a set of rules for defining data markup in a plain text format.

XSLT

Extensible Stylesheet Language transformations. XSLT describes how to transform the structure of an XML document into a differently-structured XML document. XSLT is an extension of the Extensible Stylesheet Language (XSL). XSLT shows how to reorganize the XML document into another data structure (that can then be presented by following an XSL style sheet).

Index

A

- activity
 - populating a business process with, 7-8
- adapter.ini file
 - configuring for D3L, B-28
 - setting the ota.d3ls parameter, B-6, B-28, B-44
 - setting the ota.type parameter, B-28, B-44
 - specifying the default message endpoint, B-25
- Adapters, 1-2, 10-8
- adapters
 - agents, bridges, 9-2
 - D3L responsibilities, B-1
 - integration, 1-8
 - sdk, 1-3
- Additional Parameters for RAC Configuration, 9-9
- agents, 9-2
 - use with D3L, B-23
- application data types, 3-1
 - importing a D3L file, B-29
- application view, 3-1
- applications, 2-2
 - application view, 3-1
 - creating, 3-1
 - overview, 3-1
- attributes
 - adding to common data types, 3-3
 - deleting and clearing from common data types, 3-5
 - importing for common data types, 3-4
 - modifying mappings, 6-7
 - removing mappings, 6-7
- Auto Deployment, 8-2

B

- bridges, 9-2
 - use with D3L, B-23
- browsers.init file
 - configuring for D3L, B-27
- business objects, 2-2
 - creating, 3-2
 - defining in a D3L file, B-9
 - overview, 3-2
- business process, 7-7
 - creating, 7-8

- populating with activities, 7-8

C

- common data types
 - adding attributes, 3-3
 - creating, 3-2
 - deleting and clearing attributes, 3-5
 - importing a D3L file, B-29
 - importing attributes, 3-4
- common view, 2-2
 - defining, 3-2
 - overview, 3-2
- components, 1-2
 - oracle workflow, 1-3
 - OracleAS interconnect hub, 1-2
 - repository, 9-3
 - sdk, 1-3
- content-based routing, 2-4
 - working with, 6-2
- context menu, 2-9
- cross reference tables, 2-5
 - adding applications, 6-7
 - populating, 6-7
 - removing applications, 6-7
 - working with, 6-7

D

- D3L. *See* Data Definition Description Language (D3L)
- Data Definition Description Language (D3L)
 - common view, B-31
 - common view to native format message outgoing messages translations, B-25
 - configuration
 - configuring a native format message with a D3L file, B-5, B-28
 - configuring the browsers.init file with iStudio, B-27
 - configuring with OracleAS InterConnect Adapters, B-28
 - creating a D3L file describing native format messages, B-27
 - creating a native format message, B-27
 - defining metadata properties with each event, B-30

- importing a D3L file in iStudio, B-29
- D3L DTD, B-52
- D3L file examples, B-5, B-8, B-27, B-33, B-49, B-50, B-51
- defining business objects, B-9
- defining events, B-9
- definition, B-1
- file structure example, B-8
- installing, B-26
- iStudio values in the D3L file, B-9
- magic value message header attributes, B-7, B-24
- message header attributes, B-6
- name/value pair message header attributes, B-6, B-24, B-28
- native format message examples, B-5, B-27, B-32, B-50, B-51
- native format message to common view incoming message translations, B-24
- runtime initialization, B-23
- setting the ota.d3ls parameter in the adapter.ini file, B-6
- supported data types, B-10
- unsuitable D3L formats, B-2
- use case, B-31
 - configuring the aqapp_pub and fileapp_sub applications in iStudio, B-34
 - creating a business object Employee, B-35
 - creating a D3L file for the FTP adapter, B-32
 - creating a DTD file for the Advanced Queuing Adapter, B-32
 - creating a new workspace and new project, B-34
 - creating the aqapp_pub application, B-36
 - creating the fileapp_sub application, B-40
 - creating the newEmployee Event, B-35
 - defining the application queue for the aqapp_pub application, B-39
 - defining the application view, B-37, B-40
 - defining the application view to common view mapping, B-38, B-41
 - enabling the aqapp_pub application to publish the newEmployee event, B-36
 - installing the Advanced Queuing and FTP adapters, B-42
 - making the fileapp_sub application subscribe to the newEmployee event, B-40
 - overview of aqapp_pub and fileapp_sub applications, B-31
 - running the use case, B-44
 - selecting the event to publish, B-36
 - selecting the event to which to subscribe, B-40
 - using other adapters, B-47
 - using XML mode, B-47
- when to use, B-2
- XML mode, B-2
- data types
 - supported by D3L, B-10
- Delete a Range of Messages, 10-5
- Delete a Single Message, 10-5
- Delete All Messages From a Sender, 10-5

- Delete All Messages Targeted To a Receiver, 10-5
- Delete Messages, 10-4
- deploy navigation tree, 2-9
- Deploying to Oracle Workflow, 7-9
- design navigation tree, 2-9
- design time, 1-5
- design time tools, 7-3
- detail view, 2-10
- development kit, 1-3
- domain value mapping tables
 - deleting, 6-6
- domain value mappings, 2-5
 - adding applications to, 6-5
 - deleting, 6-6
 - modifying data in, 6-6
 - removing, 6-5
 - working with, 6-5

E

- edit menu, 2-7
- enabling infrastructure, 6-1
 - content-based routing, 6-2
 - cross reference tables, 6-7
 - domain value mappings, 6-5
- Error Management, 10-7
- event maps, 4-2
- event menu, 2-7
- events, 2-2
 - creating, 4-2
 - defining in a D3L file, B-9
 - importing a D3L file, B-29
 - publishing, 4-3
 - subscribing, 4-8
- Export messages to a file, 10-5

F

- features
 - integration lifecycle management, 1-8
 - integration logic, platform functionality, 1-6
 - integration methodology, 1-6
 - message delivery, 9-5
 - message retention, 9-5
 - messaging paradigms, 9-4
 - routing support, 9-5
 - standard messaging, 1-4

H

- help menu, 2-8
- Hub, 10-2
- hub and spoke
 - how it works, 1-6
- Hub Queue Management, 10-3

I

- Import messages from a file, 10-6
- infrastructure
 - enabling, 6-1

- integration
 - create a cross reference table, A-8
 - create a project, A-4
 - create an oracle workflow process bundle, A-18
 - create applications, A-7
 - create business object events, A-5
 - create common view business object, A-5
 - create content based routing, A-17
 - create publish events, A-8
 - creating objects in oracle workflow for modeling, A-20
 - deploy the process bundle to oracle workflow, A-19
 - deployment, A-25
 - dtd code, A-6
 - exporting and installing code, A-27
 - implementing the scenario, A-3
 - legacy system, A-1
 - legacy system database trigger, A-4
 - modeling business logic in oracle workflow, A-23
 - modeling the integration, A-2
 - new centralized system, A-1
 - overview, A-1
 - setting queues, A-25
 - subscribing to events, A-11
 - synching adapters, A-26
 - the integration scenario, A-2
 - using adapters, 1-8
- integration architecture, 9-1
- integration logic, 1-6
- integration methodology, 1-6
- integration process
 - design time, 1-5
 - overview, 1-5
 - runtime, 1-6
- InterConnect Manager, 10-1
- istudio, 1-3
 - activities, 7-7
 - adding applications to cross reference tables, 6-7
 - adding applications to domain value mappings, 6-5
 - application data types, 3-1
 - applications, 2-2
 - common views and business objects, 2-2
 - concepts, 2-1
 - content-based routing, 2-4, 6-2
 - context menu, 2-9
 - creating a business object, 3-2
 - creating a business process, 7-8
 - creating a new project, 2-11
 - creating a new workspace, 2-10
 - creating a procedure, 5-2
 - creating a process bundle, 7-8
 - creating an application, 3-1
 - creating an event, 4-2
 - creating common data types, 3-2
 - cross reference tables, 2-5, 6-7
 - deleting domain value mapping tables, 6-6
 - deleting domain value mappings, 6-6
 - deploy navigation tree, 2-9
 - design navigation tree, 2-9
 - detail view, 2-10
 - domain value mapping, 2-5
 - domain value mappings, 6-5
 - event maps, 4-2
 - events, 2-2
 - exporting stored procedures, 8-1
 - implementing a procedure, 5-6
 - invoking a procedure, 5-3
 - invoking and implementing a procedure, 5-3
 - launching oracle workflow builder, 7-12
 - launching oracle workflow tools, 7-11
 - launching the oracle workflow home page, 7-12
 - mapping, transformations, 2-3
 - menu bar, 2-6
 - metadata versioning, 2-4
 - modifying attribute mappings, 6-7
 - modifying data in domain value mappings, 6-6
 - opening a project, 2-12
 - opening a workspace, 2-10
 - overview, 2-1
 - parts of the window, 2-5
 - populating cross reference tables, 6-7
 - procedures, 2-3, 5-1
 - projects, 2-11
 - publishing an event, 4-3
 - removing applications from cross reference tables, 6-7
 - removing applications from domain value mappings, 6-5
 - removing attribute mappings, 6-7
 - routing, message capability matrix, 2-5
 - sdk, 1-3
 - starting, 2-5
 - subscribing to an event, 4-8
 - toolbar, 2-8
 - tracking fields, 2-4
 - workspaces, 2-10

L

List Messages, 10-3, 10-7

M

Manual Deployment, 8-1

- mapping, 2-3
- mapping and transformations, 2-3
- menu bar, 2-6
 - edit menu, 2-7
 - event menu, 2-7
 - file menu, 2-6
 - help menu, 2-8
 - procedure menu, 2-7
- message capability matrix, 2-5
- Message Tracking, 10-12
- Message-based partitioning, 9-6
- messaging
 - standard, 1-4
 - supported paradigms, 1-4

metadata
 defining for D3L, B-30
metadata versioning, 2-4
Model Business Process, 7-6

N

native format message
 examples, B-5, B-27, B-32, B-50, B-51

O

oracle workflow, 1-3, 7-1
 apply business logic, 7-6
 composite services, 7-2
 deploy business process for runtime, 7-6
 design business process, 7-6
 design time tools, 7-3
 error management, 7-1
 Human Interaction, 7-2
 integration with oracle applications
 interconnect, 7-3
 launching oracle workflow builder, 7-12
 launching the home page, 7-12
 launching tools, 7-11
 message junctions, 7-2
 modify existing processes, 7-13
 overview, 7-1
 runtime, 7-5
 solves business problems, 7-1
 stateful routing, 7-2
OracleAS InterConnect, C-1
OracleAS interconnect
 components, 1-2
 overview, 1-1
 sdk, 1-3
 using oracle workflow, 7-6
OracleAS interconnect components
 development kit, 1-3
OracleAS interconnect hub, 1-2
OracleAS InterConnect iStudio, 2-6
ota.d3ls parameter
 setting in the adapter.ini file, B-6, B-28, B-44
ota.type parameter
 setting in the adapter.ini file, B-2, B-28, B-44

P

platform functionality, 1-6
procedure menu, 2-7
procedures, 2-3
 creating, 5-2
 exporting stored procedures, 8-1
 implementing, 5-6
 importing a D3L file, B-29
 invoking, 5-3
 invoking and implementing, 5-3
 using, 5-1
process bundle, 7-6
 creating, 7-8
projects

 creating, 2-11
 opening, 2-12
 using, 2-11
Pure Load Balancing partitioning, 9-6

R

RAC, 9-7
 configuration, 9-8
Real Application Clusters, 9-7
Repositories, 10-11
Resend Messages, 10-7
routing, 2-5
routing support
 content-based routing, 9-5
runtime, 1-6, 7-5
 components, 9-2
 features, 9-4
runtime components
 adapters, 9-2
 advanced queues, 9-4
 workflow, 9-4

S

standard messaging, 1-4
stored procedures, 8-1
Sync Adapters from iStudio, 8-7

T

toolbar, 2-8
tracking fields, 2-4
transformations, 2-3
 char replace, C-4
 concat fields, C-1
 conditional copy, C-2
 copy object, C-1
 delete xref, C-6
 expand fields, C-1
 false conditional concat, C-3
 false conditional copy, C-3
 false conditional to number, C-4
 increment, C-7
 l trim, C-5
 lookup dvm, C-6
 lookup xref, C-6
 lpad, C-5
 r trim, C-5
 rpad, C-6
 set constant, C-2
 string replace, C-5
 to number, C-4
 true conditional concat, C-3
 true conditional copy, C-3
 true conditional lookup dvm, C-2
 true conditional lookup xref, C-2
 true conditional to number, C-3
 truncate, C-7
 x substring, C-4

W

workspaces

 creating, 2-10

 opening, 2-10

 using, 2-10

