

# **Oracle® XML Gateway**

User's Guide

Release 11*i*

**Part No. B10665-03**

July 2005

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.

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.

Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

---

# Contents

## Send Us Your Comments

## Preface

## 1 Oracle XML Gateway Overview

<b>Overview of XML Gateway</b> . . . . .	1-1
XML Standards . . . . .	1-1
XML Gateway Introduction . . . . .	1-1
XML Gateway Features . . . . .	1-2
XML Gateway Architecture . . . . .	1-3
XML Messages and EDI Transactions . . . . .	1-8
Conclusion . . . . .	1-9

## 2 Message Designer

<b>Message Designer Overview</b> . . . . .	2-1
Message Designer Menus . . . . .	2-2
Message Designer Toolbar . . . . .	2-3
Message Designer Buttons . . . . .	2-3
File > Properties Menu . . . . .	2-4
<b>Message Designer Wizards</b> . . . . .	2-10
Data Definition Creation Wizard Process Flow . . . . .	2-12
Map Creation Wizard Process Flow . . . . .	2-13
<b>Using the Data Definition Creation Wizard.</b> . . . . .	2-14
Data Definition Creation Wizard Steps . . . . .	2-15
Message Designer Data Definition Window . . . . .	2-17
<b>Using the Map Creation Wizard.</b> . . . . .	2-17
<b>Transaction Map Window</b> . . . . .	2-32
Source Definition . . . . .	2-33
Transaction Map - Target Definition . . . . .	2-39
Transaction Map - Level Mapping Tab . . . . .	2-44
Transaction Map - Element Mapping . . . . .	2-48
Transaction Map - Actions . . . . .	2-51
<b>Map Action Editor</b> . . . . .	2-53
Overview . . . . .	2-53
Map Action Editor - Assignments: Assign Variable Value . . . . .	2-58

Map Action Editor - Assignments: Create Global Variable . . . . .	2-58
Map Action Editor - Database Functions: Assign Next Sequence Value . . . . .	2-59
Map Action Editor - Database Functions: Append Where Clause . . . . .	2-60
Map Action Editor - Database Functions: Insert into Database Table . . . . .	2-62
Map Action Editor - Derivations: Derive Address ID from Location Code . . . . .	2-63
Map Action Editor - Derivations: Derive Parent ID from Location Code . . . . .	2-64
Map Action Editor - Function Call: Execute Function Call . . . . .	2-65
Map Action Editor - Math Functions . . . . .	2-65
OAG Conversions . . . . .	2-66
Map Action Editor - Convert to OAG DATETIME . . . . .	2-67
Map Action Editor - Convert to OAG OPERAMT . . . . .	2-67
Map Action Editor - Convert to OAG QUANTITY . . . . .	2-69
Map Action Editor - Convert to OAG AMOUNT . . . . .	2-70
Map Action Editor - Convert from OAG DATETIME . . . . .	2-71
Map Action Editor - Convert from OAG OPERAMT . . . . .	2-72
Map Action Editor - Convert from OAG Quantity . . . . .	2-73
Map Action Editor - Convert from OAG AMOUNT . . . . .	2-74
Map Action Editor - Other: Exit Program . . . . .	2-75
Map Action Editor - Get Predefined Variable Value . . . . .	2-76
Map Action Editor - Procedure Call: Execute Procedure . . . . .	2-78
Map Action Editor - Return Error Message: Send Error Message . . . . .	2-81
Map Action Editor - String Functions: Perform Concatenation . . . . .	2-82
Map Action Editor - String Functions: Perform Substring . . . . .	2-83
Map Action Editor - XSLT Transformation . . . . .	2-84
<b>How to Extend DTDs . . . . .</b>	<b>2-84</b>
<b>How to Map a Pass-Through Message . . . . .</b>	<b>2-86</b>
<b>How to Map to an API . . . . .</b>	<b>2-86</b>
<b>Loading and Deleting Message Maps and DTDs . . . . .</b>	<b>2-87</b>
Loading/Deleting a Map . . . . .	2-87
Loading and Deleting a DTD . . . . .	2-88
<b>Downloading a Map . . . . .</b>	<b>2-88</b>
<b>Loading and Deleting an XSLT Style Sheet . . . . .</b>	<b>2-89</b>
<b>How to Implement Attachments in XML Messages . . . . .</b>	<b>2-90</b>
Attachments and Oracle E-Business Suite . . . . .	2-90
Attachments and OAG Standard . . . . .	2-90
Attachments and Oracle Transport Agent (OTA) . . . . .	2-91
Attachments and Outbound Documents. . . . .	2-91
Enable Attachments for Unit Test for Outbound Documents . . . . .	2-92
Attachments and Inbound Documents . . . . .	2-92
Enable Attachments for Unit Test for Inbound Documents . . . . .	2-93

### 3 XML Gateway Setup

Setup Overview . . . . .	3-1
Define System Profile Options . . . . .	3-2
Assign XML Gateway Responsibility . . . . .	3-4

<b>Define UTL_FILE_DIR Parameter</b> . . . . .	3-4
<b>Hub Definitions Form</b> . . . . .	3-5
<b>Define XML Standards Form</b> . . . . .	3-6
<b>Define Transactions Form</b> . . . . .	3-7
Define Transactions Form Fields . . . . .	3-8
Transaction Type and Transaction Subtype Naming Conventions . . . . .	3-9
Setting VERB and NOUN in OAG Standards. . . . .	3-11
<b>Define Lookup Values</b> . . . . .	3-12
<b>Trading Partner Setup</b> . . . . .	3-13
Required Communications Data . . . . .	3-19
Static and Dynamic Routing . . . . .	3-20
Trading Partner User Security . . . . .	3-24
<b>Code Conversion</b> . . . . .	3-26
Code Categories . . . . .	3-29
Accessing the Code Conversion Values . . . . .	3-29
Standard Code Conversion Form . . . . .	3-32
<b>Trading Partner Code Conversion Form</b> . . . . .	3-34

## 4 Execution Engine

<b>Execution Engine Overview</b> . . . . .	4-1
<b>Protocol Type</b> . . . . .	4-4
<b>XML Gateway Envelope</b> . . . . .	4-6
<b>Trading Partner Validation for Inbound Messages</b> . . . . .	4-9
Validation Against Data in the Trading Partner Setup Form . . . . .	4-11
<b>Trading Partner Validation for Outbound Messages</b> . . . . .	4-12
<b>How to Implement the OAG Confirmation Business Object Document</b> . . . . .	4-13
Purpose of the Confirmation Message. . . . .	4-13
Structure of the Confirmation Message . . . . .	4-13
XML Gateway Seeded Confirmation Message Maps . . . . .	4-13
E-Business Suite Seeded Events and Event Subscriptions . . . . .	4-13
How to Implement or Disable a Seeded Confirmation Message. . . . .	4-14

## 5 Message Queues

<b>Queues</b> . . . . .	5-1
Outbound Queues . . . . .	5-2
Inbound Queues . . . . .	5-2
Oracle Transport Agent Send Inbound HTML Page . . . . .	5-3
XML Gateway Message Format . . . . .	5-4

## 6 Integrating Oracle XML Gateway with Oracle Workflow Business Event System

<b>Integrating Oracle XML Gateway with Oracle Workflow Business Event System</b> . . . . .	6-1
Overview . . . . .	6-1
<b>Oracle Workflow Builder - Item Types</b> . . . . .	6-3
Components of an Item Type . . . . .	6-4

XML Gateway Standard Item Type . . . . .	6-4
XML Gateway Error Processing Item Type. . . . .	6-5
E-Business Suite Application Module-Specific Item Type . . . . .	6-5
<b>XML Gateway Standard Item Type . . . . .</b>	<b>6-5</b>
Attributes . . . . .	6-6
Processes . . . . .	6-6
Functions . . . . .	6-7
Events . . . . .	6-19
Lookup Types . . . . .	6-22
<b>XML Gateway Error Processing Item Type . . . . .</b>	<b>6-24</b>
Attributes . . . . .	6-25
Processes . . . . .	6-25
Notifications. . . . .	6-35
Functions . . . . .	6-36
Events . . . . .	6-38
Messages . . . . .	6-40
Lookup Types . . . . .	6-43
<b>Configure Oracle Prebuilt Inbound Messages . . . . .</b>	<b>6-44</b>
<b>Configure Oracle Prebuilt Outbound Messages . . . . .</b>	<b>6-45</b>
<b>Application to Application Integration . . . . .</b>	<b>6-48</b>
Inbound Option . . . . .	6-48
Outbound Option . . . . .	6-49
<b>Manage Workflow Processes . . . . .</b>	<b>6-49</b>
Register New Business Events and Event Subscription . . . . .	6-49
Identify Seeded Item Types . . . . .	6-50
Identify Seeded Business Events and Associated Event Subscriptions . . . . .	6-50
Configure or Delete Seeded Event Subscriptions . . . . .	6-50
View and Respond to Error Notifications . . . . .	6-50
Purge XML Gateway Transactions . . . . .	6-50
<b>Monitor Workflow Processes . . . . .</b>	<b>6-51</b>
Transaction-Level Trace . . . . .	6-51
Monitor Transaction Status . . . . .	6-52
Review XML Message Returned by Generate Functions . . . . .	6-52
Start Agent Listeners . . . . .	6-52
<b>Development Guidelines for Custom Messages for B2B Integration . . . . .</b>	<b>6-53</b>
Development Guidelines for Outbound Messages . . . . .	6-53
Development Guidelines for Inbound Messages . . . . .	6-54
<b>Common Questions . . . . .</b>	<b>6-54</b>
<b>Message Delivery Status . . . . .</b>	<b>6-55</b>
How Other Messaging Systems Use XML Message Delivery Callback . . . . .	6-57

## 7 Oracle Transport Agent

<b>Oracle Transport Agent Overview . . . . .</b>	<b>7-1</b>
OTA Message Propagation Flow . . . . .	7-2
Oracle Transport Agent Post Message . . . . .	7-3

Oracle Transport Agent Response Message . . . . .	7-8
OTA and Attachments . . . . .	7-9
<b>Authentication Methods</b> . . . . .	7-9
Implementation of Client Authentication . . . . .	7-9
Sequence of Events . . . . .	7-10
<b>Enabling Client Authentication</b> . . . . .	7-11
<b>Setup Parameters</b> . . . . .	7-11
Parameters Set Through AutoConfig . . . . .	7-11
Parameters Not Set Through AutoConfig . . . . .	7-12
<b>Connecting to Non-OTA Servers</b> . . . . .	7-13
<b>Code Connection Samples</b> . . . . .	7-14
<b>Troubleshooting</b> . . . . .	7-22
HTTP Status Codes . . . . .	7-22

## 8 Web Services

<b>Web Services</b> . . . . .	8-1
Web Services Components and Features . . . . .	8-1
<b>Process Flows</b> . . . . .	8-3
Inbound Process Flow . . . . .	8-3
Outbound Process Flow . . . . .	8-3
<b>Setup Steps</b> . . . . .	8-4
<b>Diagnostics</b> . . . . .	8-6
<b>Example Purchase Order Inbound Web Service</b> . . . . .	8-7
Prerequisite Files . . . . .	8-7
Example Scripts and Files . . . . .	8-8
Prerequisite Files . . . . .	8-9

## 9 XML Gateway B2B Transactions Using JMS Queues

Java Messaging Service (JMS) Overview . . . . .	9-1
Oracle XML Gateway and B2B Transactions Integration Points . . . . .	9-2
Oracle XML Gateway and JMS Integration . . . . .	9-3
Integration Features . . . . .	9-3
JMS Inbound and Outbound Messages . . . . .	9-3
Impacts on the Trading Partner Setup and Seed Data . . . . .	9-4
<b>Creating Custom JMS Queues</b> . . . . .	9-5
<b>Steps to Create Custom JMS Queues for Outbound B2B Transactions</b> . . . . .	9-5
<b>Steps to Create Custom JMS Queues for Inbound B2B Transactions</b> . . . . .	9-12

## A Map Analysis Guidelines

<b>Map Analysis Overview</b> . . . . .	A-1
<b>Map Analysis Guidelines for Outbound Messages</b> . . . . .	A-1
Map Analysis Guidelines for Outbound Messages Checklist . . . . .	A-1
Compare Database Views (Source) to DTD (Target) . . . . .	A-3
<b>Map Analysis Guidelines for Inbound Messages</b> . . . . .	A-7

Map Analysis Guidelines for Inbound Messages Checklist . . . . .	A-7
Compare Database Views (Source) to DTD (Target) . . . . .	A-9
<b>Identifying Source and Target Document Levels . . . . .</b>	<b>A-13</b>
Collapsing Levels . . . . .	A-14
Expanding Levels . . . . .	A-15
<b>Recommending DTD Additions or Changes to OAG . . . . .</b>	<b>A-16</b>
<b>Special Considerations for Custom Messages. . . . .</b>	<b>A-17</b>

## **B Seeded Code Categories**

XML Gateway Seeded Code Categories . . . . .	B-1
--	-----

## **C Supported Actions**

XML Gateway Supported Actions . . . . .	C-1
---	-----

## **D Naming Conventions**

XML Gateway Naming Conventions Summary . . . . .	D-1
--	-----

## **E Timezone Values**

XML Gateway Valid Time Zone Values . . . . .	E-1
--	-----

## **F APIs**

XML Gateway APIs . . . . .	F-1
Execution Engine APIs . . . . .	F-1
APIs Defined in ECX_STANDARD . . . . .	F-1
APIs Defined in ECX_ERRORLOG . . . . .	F-2
Message Designer APIs . . . . .	F-3
APIs Defined in ECX_STANDARD . . . . .	F-5
APIs Defined in ECX_DOCUMENT . . . . .	F-8
APIs Defined in ECX_CONDITIONS . . . . .	F-11
APIs Defined in ECX_TRADING_PARTNER_PVT . . . . .	F-12
APIs Defined in ECX_ERRORLOG . . . . .	F-17
APIs Defined in ECX_ACTIONS . . . . .	F-19
APIs Defined in ECX_ATTACHMENT . . . . .	F-20
APIs Defined in ECX_ENG_UTILS . . . . .	F-25

## **G Troubleshooting**

<b>Troubleshooting Your XML Gateway Installation . . . . .</b>	<b>G-1</b>
Automated Troubleshooting Script . . . . .	G-1
<b>Transaction Monitor . . . . .</b>	<b>G-5</b>
Transaction Monitor Search Page . . . . .	G-5
Search Results Page . . . . .	G-7
Transaction Monitor Details Page. . . . .	G-9
<b>Manual Troubleshooting Steps . . . . .</b>	<b>G-11</b>



XML Gateway Engine-Level Messages . . . . .	G-11
XML Gateway API-Level Messages . . . . .	G-26
<b>XML Gateway Version Validation . . . . .</b>	<b>G-31</b>
<b>Common Client Authentication Implementation Issues . . . . .</b>	<b>G-33</b>
<b>Oracle Diagnostic Tests . . . . .</b>	<b>G-34</b>
XML Gateway Tests . . . . .	G-34

## Index



---

# Send Us Your Comments

**Oracle XML Gateway User's Guide, Release 11*i***

**Part No. B10665-03**

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: [appsdoc\\_us@oracle.com](mailto:appsdoc_us@oracle.com)
- FAX: 650-506-7200 Attn: Oracle Applications Technology Group Documentation Manager
- Postal service:  
Oracle Applications Technology Group Documentation Manager  
Oracle Corporation  
500 Oracle Parkway  
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

## Intended Audience

Welcome to Release 11i of the *Oracle XML Gateway User's Guide*.

This guide assumes you have a working knowledge of the following:

- The principles and customary practices of your business area.
- Oracle XML Gateway.

If you have never used Oracle XML Gateway, Oracle suggests you attend training classes available through Oracle University.

- The Oracle Applications graphical user interface.

To learn more about the Oracle Applications graphical user interface, read the *Oracle Applications User's Guide*.

See Related Documents on page xiv for more Oracle Applications product information.

## TTY Access to Oracle Support Services

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

## Documentation Accessibility

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

## Accessibility of Code Examples in Documentation

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

## Accessibility of Links to External Web Sites in Documentation

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

## Structure

### **1 Oracle XML Gateway Overview**

This chapter introduces the Oracle XML Gateway. Specifically, this chapter describes the

- Oracle XML Gateway features
- Oracle XML Gateway Architecture
- Oracle XML Gateway components: Message Designer, XML Gateway Setup, and Execution Engine

### **2 Message Designer**

This chapter describes how to use the XML Gateway Message Designer to create new XML message maps or to modify Oracle prebuilt XML message maps.

### **3 XML Gateway Setup**

This chapter explains the implementation steps required to set up XML Gateway. Topics include descriptions of all required forms as well as conceptual discussions of setup options.

### **4 Execution Engine**

### **5 Message Queues**

This chapter introduces the message queues.

### **6 Integrating Oracle XML Gateway with Oracle Workflow Business Event System**

### **7 Oracle Transport Agent**

### **8 Web Services**

### **9 XML Gateway B2B Transactions Using JMS Queues**

This chapter describes how XML Gateway can be integrated with any Java Messaging Service (JMS) queues so that the JMS providers can directly publish and subscribe messages to these JMS queues and leverage the Business-to-Business (B2B) transaction processing.

### **A Map Analysis Guidelines**

### **B Seeded Code Categories**

### **C Supported Actions**

### **D Naming Conventions**

### **E Timezone Values**

This appendix lists the valid time zone values for the profile option ECX: Server Time Zone.

### **F APIs**

### **G Troubleshooting**

## Related Documents

You can choose from many sources of information, including online documentation, training, and support services, to increase your knowledge and understanding of Oracle XML Gateway

If this guide refers you to other Oracle Applications documentation, use only the Release 11*i* versions of those guides.

## Online Documentation

If you are using the version of Oracle XML Gateway embedded in Oracle Applications, note that all Oracle Applications documentation is available online (HTML or PDF)

- **PDF Documentation** - See the Oracle Applications Documentation Library CD for current PDF documentation for your product with each release. The Oracle Applications Documentation Library is also available on *OracleMetaLink* and is updated frequently.
- **Online Help** - Online help patches (HTML) are available on *OracleMetaLink*.
- **About Documents** - Refer to the About document for the mini-pack or family pack that you have installed to learn about feature updates, installation information, and new documentation or documentation patches that you can download. About documents are available on *OracleMetaLink*.

If you are using the standalone version of Oracle XML Gateway, note that this guide is available online in HTML format. The HTML documentation is available from a URL provided by your system administrator or from the help icon in the Oracle XML Gateway Web pages.

## Related Guides

You may want to refer to other Oracle Applications implementation documentation when you set up and use Oracle XML Gateway. Additionally, Oracle XML Gateway is used by other Oracle Applications products. Therefore, if you are using the version of Oracle XML Gateway embedded in Oracle Applications, you may want to refer to other products' guides to learn more about product specific information.

You can read the guides online by choosing Library from the expandable menu on your Oracle Applications HTML help window, by reading from the Oracle Applications Documentation Library CD included in your media pack, or by using a Web browser with a URL that your system administrator provides.

If you require printed guides, you can purchase them from the Oracle Store at <http://oraclestore.oracle.com>

## Guides Related to All Products

### *Oracle Applications User's Guide*

This guide explains how to enter data, query, run reports, and navigate using the graphical user interface (GUI) available with this release of Oracle Workflow (and any other Oracle Applications products). This guide also includes information on setting user profiles, as well as running and reviewing reports and concurrent processes.

You can access this user's guide online by choosing "Getting Started with Oracle Applications" from any Oracle Applications help file.

## Guides Related to This Product

### *Oracle Workflow Administrator's Guide*

This guide explains how to complete the setup steps necessary for any product that includes workflow-enabled processes, as well as how to monitor the progress of runtime workflow processes.

#### ***Oracle Workflow Developer's Guide***

This guide explains how to define new workflow business processes and customize existing Oracle Applications-embedded workflow processes. It also describes how to define and customize business events and event subscriptions.

#### ***Oracle Workflow API Reference***

This guide describes the APIs provided for developers and administrators to access Oracle Workflow.

#### ***Oracle Workflow User's Guide***

This guide describes how users can view and respond to workflow notifications and monitor the progress of their workflow processes.

#### ***Oracle e-Commerce Gateway User's Guide***

This guide describes how Oracle e-Commerce Gateway provides a means to conduct business with trading partners via Electronic Data Interchange (EDI). Data files are exchanged in a standard format to minimize manual effort, speed data processing and ensure accuracy.

## **Installation and System Administration**

#### ***Oracle Applications Concepts***

This guide provides an introduction to the concepts, features, technology stack, architecture, and terminology for Oracle Applications Release 11*i*. It provides a useful first book to read before an installation of Oracle Applications. This guide also introduces the concepts behind Applications-wide features such as Business Intelligence (BIS), languages and character sets, and Self-Service Web Applications.

#### ***Installing Oracle Applications***

This guide provides instructions for managing the installation of Oracle Applications products. In Release 11*i*, much of the installation process is handled using Oracle Rapid Install, which minimizes the time to install Oracle Applications, the Oracle8 technology stack, and the Oracle8*i* Server technology stack by automating many of the required steps. This guide contains instructions for using Oracle Rapid Install and lists the tasks you need to perform to finish your installation. You should use this guide in conjunction with individual product user's guides and implementation guides.

#### ***Upgrading Oracle Applications***

Refer to this guide if you are upgrading your Oracle Applications Release 10.7 or Release 11.0 products to Release 11*i*. This guide describes the upgrade process and lists database and product-specific upgrade tasks. You must be either at Release 10.7 (NCA, SmartClient, or character mode) or Release 11.0, to upgrade to Release 11*i*. You cannot upgrade to Release 11*i* directly from releases prior to 10.7.

#### ***Maintaining Oracle Applications***

Use this guide to help you run the various AD utilities, such as AutoUpgrade, Auto Patch, AD Administration, AD Controller, AD Relink, License Manager, and others. It contains how-to steps, screenshots, and other information that you need to run the AD



utilities. This guide also provides information on maintaining the Oracle applications file system and database.

### ***Oracle Applications System Administrator's Guide***

This guide provides planning and reference information for the Oracle Applications System Administrator. It contains information on how to define security, customize menus and online help, and manage concurrent processing.

### ***Oracle Applications Developer's Guide***

This guide contains the coding standards followed by the Oracle Applications development staff. It describes the Oracle Application Object Library components needed to implement the Oracle Applications user interface described in the *Oracle Applications User Interface Standards for Forms-Based Products*. It also provides information to help you build your custom Oracle Forms Developer 6i forms so that they integrate with Oracle Applications.

## **Other Implementation Documentation**

### ***Oracle Applications Product Update Notes***

Use this guide as a reference for upgrading an installation of Oracle Applications. It provides a history of the changes to individual Oracle Applications products between Release 11.0 and Release 11i. It includes new features, enhancements, and changes made to database objects, profile options, and seed data for this interval.

### ***Multiple Reporting Currencies in Oracle Applications***

If you use the Multiple Reporting Currencies feature to record transactions in more than one currency, use this manual before implementing Oracle Alert. This manual details additional steps and setup considerations for implementing Oracle Alert with this feature.

### ***Multiple Organizations in Oracle Applications***

This guide describes how to set up and use Oracle Alert with Oracle Applications' Multiple Organization support feature, so you can define and support different organization structures when running a single installation of Oracle Alert.

### ***Oracle Workflow Guide***

This guide explains how to define new workflow business processes as well as customize existing Oracle Applications-embedded workflow processes. You also use this guide to complete the setup steps necessary for any Oracle Applications product that includes workflow-enabled processes.

### ***Oracle Applications Flexfields Guide***

This guide provides flexfields planning, setup and reference information for the Oracle Applications implementation team, as well as for users responsible for the ongoing maintenance of Oracle Applications product data. This manual also provides information on creating custom reports on flexfields data.

### ***Oracle eTechnical Reference Manuals***

Each eTechnical Reference Manual (eTRM) contains database diagrams and a detailed description of database tables, forms, reports, and programs for a specific Oracle Applications product. This information helps you convert data from your existing

applications, integrate Oracle Applications data with non-Oracle applications, and write custom reports for Oracle Applications products. Oracle eTRM is available on *Metalink*.

#### ***Oracle Applications User Interface Standards for Forms-Based Products***

This guide contains the user interface (UI) standards followed by the Oracle Applications development staff. It describes the UI for the Oracle Applications products and how to apply this UI to the design of an application built by using Oracle Forms.

#### ***Oracle Manufacturing APIs and Open Interfaces Manual***

This manual contains up-to-date information about integrating with other Oracle Manufacturing applications and with your other systems. This documentation includes APIs and open interfaces found in Oracle Manufacturing.

#### ***Oracle Order Management Suite APIs and Open Interfaces Manual***

This manual contains up-to-date information about integrating with other Oracle Manufacturing applications and with your other systems. This documentation includes APIs and open interfaces found in Oracle Order Management Suite.

#### ***Oracle Applications Message Reference Manual***

This manual describes all Oracle Applications messages. This manual is available in HTML format on the documentation CD-ROM for Release 11*i*.

## **Training and Support**

### **Training**

Oracle offers a complete set of training courses to help you and your staff master Oracle XML Gateway and reach full productivity quickly. These courses are organized into functional learning paths, so you take only those courses appropriate to your job or area of responsibility.

You have a choice of educational environments. You can attend courses offered by Oracle University at any one of our many Education Centers, you can arrange for our trainers to teach at your facility, or you can use Oracle Learning Network (OLN), Oracle University's online education utility. In addition, Oracle training professionals can tailor standard courses or develop custom courses to meet your needs. For example, you may want to use your organization's structure, terminology, and data as examples in a customized training session delivered at your own facility.

### **Support**

From on-site support to central support, our team of experienced professionals provides the help and information you need to keep Oracle XML Gateway working for you. This team includes your Technical Representative, Account Manager, and Oracle's large staff of consultants and support specialists, with expertise in your business area, managing an Oracle Database, and your hardware and software environment.

## **Do Not Use Database Tools to Modify Oracle Applications Data**

Oracle **STRONGLY RECOMMENDS** that you never use SQL\*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle Applications data unless otherwise instructed.

Oracle provides powerful tools you can use to create, store, change, retrieve, and maintain information in an Oracle database. But if you use Oracle tools such as SQL\*Plus

to modify Oracle Applications data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle Applications tables are interrelated, any change you make using an Oracle Applications form can update many tables at once. But when you modify Oracle Applications data using anything other than Oracle Applications, you may change a row in one table without making corresponding changes in related tables. If your tables get out of synchronization with each other, you risk retrieving erroneous information and you risk unpredictable results throughout Oracle Applications.

When you use Oracle Applications to modify your data, Oracle Applications automatically checks that your changes are valid. Oracle Applications also keeps track of who changes information. If you enter information into database tables using database tools, you may store invalid information. You also lose the ability to track who has changed your information because SQL\*Plus and other database tools do not keep a record of changes.



---

# Oracle XML Gateway Overview

This chapter introduces the Oracle XML Gateway. Specifically, this chapter describes the

- Oracle XML Gateway features
- Oracle XML Gateway Architecture
- Oracle XML Gateway components: Message Designer, XML Gateway Setup, and Execution Engine

This chapter covers the following topics:

- Overview of XML Gateway

## Overview of XML Gateway

### XML Standards

Many standards bodies (for example, OAG, Rosettanet, SOAP, and iFX) exist with published Document Type Definitions (DTDs). Some standards are strong at managing the message content while others excel at managing both the message content and its related processes.

As a provider of broad-based business application solutions to support all industries, Oracle XML Gateway supports all DTD based XML standards. The majority of messages delivered with the Oracle E-Business Suite are mapped using the Open Application Group (OAG) standard. Any prebuilt message can be remapped to any standard of choice using the XML Gateway Message Designer.

### XML Gateway Introduction

Oracle XML Gateway is a set of services that allows easy integration with the Oracle E-Business Suite to support XML messaging.

The Oracle E-Business Suite utilizes the Oracle Workflow Business Event System to support event-based XML message creation and consumption.

Oracle XML Gateway consumes events raised by the Oracle E-Business Suite and subscribes to inbound events for processing. Oracle XML Gateway uses the message propagation feature of Oracle Advanced Queuing to integrate with the Oracle Transport Agent to deliver messages to and receive messages from business partners.

Oracle XML Gateway supports both Business-to-Business (B2B) and Application-to-Application (A2A) initiatives. B2B initiatives include communicating

business documents and participating in industry exchanges. An example of an A2A initiative is data integration with legacy and disparate systems.

With Oracle XML Gateway services, you are assured consistent XML message implementation when integrating with the Oracle E-Business Suite, thereby lowering integration costs and expediting message implementation while supporting corporate e-business initiatives.

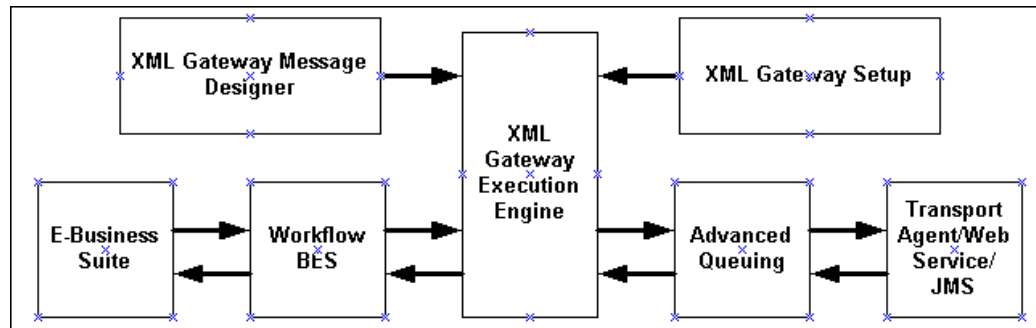
## XML Gateway Features

The Oracle XML Gateway offers the following features and integrations:

- **Message Designer**  
Use this wizard-guided tool to define message maps containing the RDBMS-to-XML or XML-to-XML data transformations.
- **Robust Execution Engine**  
The execution engine integrates with the Oracle Workflow Business Event System for event-based message creation and consumption.
- **Flexible Trading Partner Directory Service**  
Use this service to define a hub, all trading partners exchanging on a hub, or a specific business partner. The definition includes information on enabled messages, confirmation requests, and message maps and message transport protocols.
- **Flexible message setup**  
Flexible conversion of Oracle codes to recipient or standards-based codes, including a cross-reference between the Oracle internal and external transaction names.
- **Integration with Oracle Workflow Error Handling**  
Oracle Workflow error handling process provides active error notification to the XML Gateway system administrator or Trading Partner contact with support for "retry" and "reprocess" for failed processes.
- **Integration with Oracle Advanced Queuing**  
Oracle AQ provides persistent storage and message propagation.
- **Utilizes Servlet-Based Transport Agent**  
The servlet-based Transport Agent delivers and receives XML messages using SMTP, HTTP, or HTTPS protocols.
- **Support for Web Services**  
In Release 11*i*, any transactions supported by Oracle XML Gateway can be sent or received as a document style Web service using the Simple Object Access Protocol (SOAP).
- **Support for Java Messaging Service (JMS)**  
To provide complete support for Business-to-Business transactions, Oracle XML Gateway leverages Oracle Workflow Business Event System to enable the exchange of JMS messages between the Oracle E-Business Suite and Trading Partners.

## XML Gateway Architecture

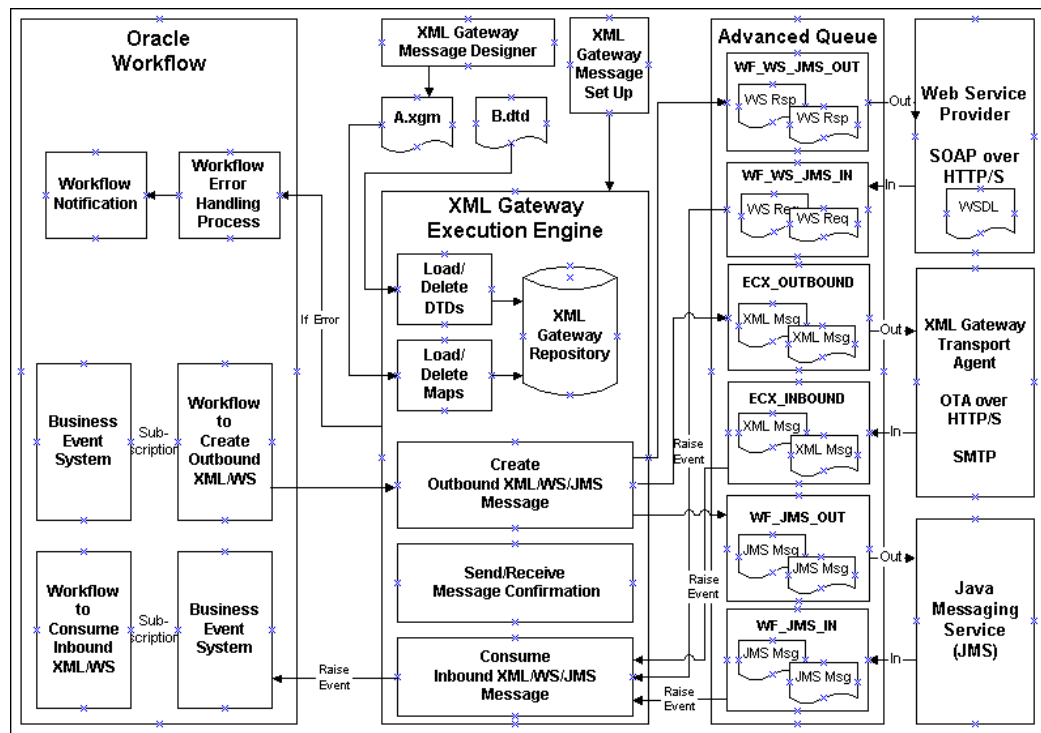
The following diagram shows an overview of the XML Gateway architecture:



The services supported by Oracle XML Gateway are grouped into four functional areas as follows:

- Message Designer
- XML Gateway Setup
- Execution Engine
- Transport Agent

The following figure shows the detailed flow of an XML message through XML Gateway:



## Message Designer

The XML Gateway Message Designer is a wizard-guided, repository-based tool used to define message maps. A message map represents the relationship between the source and target data elements.

Use the XML Gateway Message Designer to perform the following functions:

### Define Data Source and Data Target

Each message map consists of a data source and data target representing where the data is coming from and where it is mapped to. The XML Gateway Message Designer supports the following combinations of source and target data definitions:

- Source RDBMS to Target XML
- Source XML to Target RDBMS
- Source XML to Target XML

RDBMS-based data definitions can be based on database tables, database views, Application Open Interface tables, or Application APIs.

XML-based data definitions can be based on an XML Document Type Definition or a production XML message. Using a production XML message as a data source or data target is the recommended choice if you are migrating from an existing implementation or legacy system (where XML messaging is supported) to the Oracle E-Business Suite.

### Map Source Data Structure to Target Data Structure

Once the data source and data target are defined, use the Message Designer to relate the source data structure to the target data structure.

This process is especially important if the data structure of the business document is different from the data structure of the application data model. For example, if an inbound business document is represented in three levels but the application data model is represented in two levels, the data in the business document must be collapsed to accommodate the application data model. The incoming data is expanded if the opposite case occurs in which the inbound business document is represented in two levels but the application data model is represented in three levels.

The XML Gateway Message Designer supports both expanding and collapsing hierarchies to ensure that data can be retrieved from or populated into the Oracle E-Business Suite data model.

### Map Source Data Element to Target Data Element

Once the data source, data target, and source-to-target hierarchy are defined, use the Message Designer to map the source data elements to the target data elements.

The Message Designer user interface displays the data source and the data target in adjacent windows. A simple drag and drop between the source and target data elements creates a map relationship. The source data element name is noted next to the target data element name to identify the map relationship.

### Identify Data Transformation and Process Control Functions

As part of the element mapping process, use the Message Designer to identify data transformation and process control functions. These functions can be defined at the source or target as follows:

- To be applied at the data element, document, or root level



- To be applied before, during, or after the message is created or consumed

The function may be qualified by a condition. If no condition is defined, the function will always be applied. See XML Gateway Supported Actions, page C-1 for a list of the functions supported by the XML Gateway.

The common data transformation functions involve math functions and string manipulation in addition to conversions between the Oracle and OAG formats representing date, operating amount, quantity, and amount values.

The common process control functions involve calling procedures or database functions to extend the integration with the Oracle E-Business Suite. Other common process control functions allow you to inquire on the status of a transaction and manage the process flow based on the status. For serious errors, the process may be aborted with error notifications sent to the XML Gateway system administrator or Trading Partner contact.

Once the message map is created, it is loaded along with its associated DTDs into the XML Gateway repository for use by the Execution Engine to create outbound or to consume inbound XML messages.

## **XML Gateway Setup**

To implement a message with a trading partner, use the XML Gateway setup features to define the Trading Partner or hub, code conversion values, and internal-to-external transaction name cross-references. In addition, you can identify the XML Gateway system administrator to notify for system or process errors.

### **Define Trading Partner or Hub**

E-Business may be conducted directly with a business partner, commonly known as a trading partner, or via a hub, such as Oracle Exchange, where many buyers and sellers converge to conduct electronic commerce.

With Oracle XML Gateway, you can define the hub or the individual business partner as a trading partner. If you define the hub as the trading partner, you can identify all the buyers and sellers who are conducting business on the hub as Trading Partners to the hub.

Included in the Trading Partner or hub definition are the following:

- Define Trading Partner or Hub name
- Enable XML messages for the partner
- Enable request for message confirmation
- Define the message map to use for message creation or consumption
- Define the e-mail address of the trading partner contact to notify for data errors
- Define trading partner-specific code conversion values
- Define transport protocol: SMTP, HTTP, HTTPS, JMS with credential and username and password as necessary

### **Define Code Conversion**

With Oracle XML Gateway, you can cross-reference Oracle codes to codes that are meaningful to your recipient. Conversely, you can cross-reference codes you receive from your partner to codes that are meaningful to your Oracle Application. Common

examples of Oracle E-Business Suite codes requiring code conversion are units of measure and currency code.

Code conversion values can be defined to be applied universally for all Trading Partners and all messages. Additionally, code conversion values can be defined for a specific XML standard or specific to a Trading Partner.

### **Define Transactions**

Use Oracle XML Gateway to define a cross-reference between the Oracle transaction name and the external transaction name. The external transaction name will be based on what is meaningful per the XML standard used by the recipient. The external transaction name will appear on the message envelope to support message transport.

### **Execution Engine**

The XML Gateway Execution Engine is responsible for interacting with several Oracle technologies to process and transport XML messages to and from Trading Partners for B2B integration, or other information systems both within and outside the enterprise for A2A integration.

The Oracle technologies involved include the following:

- Oracle Workflow Business Event System
- Oracle E-Business Suite
- Oracle Advanced Queuing
- Oracle Workflow

### **Oracle Workflow Business Event System**

Oracle XML Gateway leverages the Oracle Workflow Business Event System to publish and subscribe to application business events of interest to automatically trigger message creation or consumption.

Business events and event subscriptions to send outbound messages or to consume inbound messages are delivered for all Oracle prebuilt messages. The seeded event subscriptions can be configured during implementation to perform activities to address specific business requirements.

### **Oracle E-Business Suite**

The XML Gateway Execution Engine interfaces with the Oracle E-Business Suite via business events and event subscriptions to retrieve data from or populate data into the Oracle e-Business Suite tables.

### **Oracle Advanced Queuing**

The XML Gateway Execution Engine interfaces with Oracle Advanced Queuing to stage outbound XML messages or receive inbound XML messages for processing.

### **Oracle Workflow**

The XML Gateway Execution Engine interfaces with Oracle Workflow to actively notify the XML Gateway system administrator regarding system or process errors, or the Trading Partner contact for data errors.

The XML Gateway system administrator has the option to "retry" failed outbound processes, or "reprocess" failed inbound processes.

## **XML Gateway Execution Engine Internal Functions:**

Internal to the XML Gateway Execution Engine are the following functions:

### **Validate Trading Partner or Hub**

Verify that the Trading Partner is defined and the required document is enabled.

### **Retrieve Message Map from Repository**

Retrieve the message map associated with the Trading Partner and required document.

### **Retrieve and Populate Data to and from the Oracle E-Business Suite**

For outbound messages, gather the application data from the Oracle E-Business Suite using the information on the message map.

For inbound messages, process the data into the Oracle E-Business Suite using the information on the message map.

### **Apply Data Transformations or Process Control Functions**

Apply code conversions, data transformations, and process control functions defined on the message map.

### **Validate Message Using the XML Parser**

For outbound message, the XML Parser is used to validate the newly created message to ensure that it is well-formed and valid. Poorly formed or invalid messages will not be enqueued for delivery. An error notification is sent to the XML Gateway system administrator.

Inbound messages dequeued from the inbound queue are validated using the XML Parser to ensure that it is well-formed and valid. Poorly formed or invalid messages will not be processed by the XML Gateway Execution Engine. An error notification is sent to the Trading Partner contact.

### **Enqueue and Dequeue XML Messages**

Outbound messages are enqueued for delivery by the transport agent. Inbound messages are dequeued for processing by the XML Gateway Execution Engine.

### **Send and Receive Message Confirmations**

An outbound confirmation is sent in response to an inbound request if the outbound confirmation is enabled for the Trading Partner.

An inbound confirmation is received in response to an outbound request if the inbound confirmation is enabled for the Trading Partner.

### **Direct Interaction with Execution Engine for A2A Integration**

Seeded Workflow functions are provided for use in Workflow processes to interact directly with the Execution Engine to generate outbound or to consume inbound messages.

The outbound message generated by the XML Gateway Execution Engine is made available to the downstream Workflow activity for processing. The Execution Engine consumes the inbound message passed to it by a Workflow process.

The XML Gateway Execution Engine begins processing once it detects the Workflow process to send or generate an XML message or that an inbound message has arrived on the inbound queue.

## **Oracle Transport Agent**

The Oracle Transport Agent interfaces with Oracle Advanced Queuing to deliver outbound messages and to receive inbound messages.

The Transport Agent server is a Java-based servlet that uses the Transport Agent Messaging Protocol to support the following:

- Guaranteed delivery, exactly once
- Message encryption
- Server-to-server certificate authentication

## Web Services

Oracle XML Gateway uses Web Services Description Language (WSDL) to inform trading partners how to communicate with the Oracle E-Business Suite. The Suite also publishes the WSDL to a URL for customers to access. Partners can use any third party Web service tools to call for Web services.

All inbound Web service messages are received through the Simple Object Accessed Protocol (SOAP) servlet running under the Web service provider. These messages are prepared as designed events before being enqueued to the SOAP agent, WF\_WS\_JMS\_IN queue, for further processing.

Outbound messages are created by Workflow processes and passed to the SOAP agent, WF\_WS\_JMS\_OUT queue. The messages are picked up by the SOAP client who is responsible for the actual delivery of the message to the Trading Partner.

## JMS Queues

Java Messaging Service (JMS) is a message standard. To send JMS messages between the Oracle E-Business Suite and Trading Partners, the appropriate Protocol Type (JMS) and Protocol Address registered with the Business Event System must be identified first in the Trading Partner Setup form.

JMS providers can integrate with XML Gateway for B2B transactions using WF\_JMS\_IN, WF\_JMS\_OUT, or their own JMS queues.

For inbound transactions, this integration provides complete validation and authorization support so that XML Gateway processes only those messages that are valid and have the appropriate authorization.

For outbound transactions, this integration provides a mechanism to store the JMS queues as part of the Trading Partner setup so that the generated messages can be sent to the desired JMS queue.

## XML Messages and EDI Transactions

EDI transactions and XML messages are two forms of electronic messaging based on their respective standards. EDI transactions are more batch-oriented while their XML counterpart is event-based, real-time, and tend to be based on a single transaction.

Oracle e-Commerce Gateway integrates with the Oracle E-Business Suite to extract or to import traditional EDI transactions using a flat ASCII file. A third party EDI Translator is required to map the data between the Oracle E-Business Suite and the EDI standard of choice such as ASC X12 or EDIFACT.

Oracle XML Gateway integrates with the Oracle E-Business Suite to create or to consume XML messages based on application business events. Oracle XML Gateway creates and consumes standards-compliant XML messages without the use of a translator.

## Conclusion

Oracle XML Gateway is the XML message enabler for the Oracle E-Business Suite. It provides consistent XML message implementation within and outside the enterprise in support of A2A and B2B messaging initiatives.

Oracle XML Gateway leverages the publish and subscribe features of the Oracle Workflow Business Event System to automate message creation and consumption in addition to using Oracle Workflow to link key business processes. This combination enables seamless collaboration, coordination, and communication of business-critical data throughout the supply chain.



---

## Message Designer

This chapter describes how to use the XML Gateway Message Designer to create new XML message maps or to modify Oracle prebuilt XML message maps.

This chapter covers the following topics:

- Message Designer Overview
- Message Designer Wizards
- Using the Data Definition Creation Wizard
- Using the Map Creation Wizard
- Transaction Map Window
- Map Action Editor
- How to Extend DTDs
- How to Map a Pass-Through Message
- How to Map to an API
- Loading and Deleting Message Maps and DTDs
- Downloading a Map
- Loading and Deleting an XSLT Style Sheet
- How to Implement Attachments in XML Messages

### Message Designer Overview

The Message Designer is a wizard-guided, repository-based tool used to create XML message maps. Map creation consists of defining the data source and data target, defining hierarchy and data maps between the source and target data, and defining actions for data transformation and process control.

The Message Designer is independent of XML standards. It complies with version 1.0 of the W3C XML specifications.

**Note:** Information regarding the W3C XML Standards can be found at <http://www.w3.org/XML/Activity>.

The Message Designer can support map creation for any business document as long as the document conforms to a Document Type Definition (DTD).

The Message Designer can be used to:

- Modify the Oracle prebuilt message maps
- Create new message maps

If you are using the Message Designer to create new message maps, you must complete a map analysis before attempting to use the Message Designer. Refer to Map Analysis Guidelines, page A-1 for the details. The thoroughness of your map analysis will impact the success of your map creation.

Load completed message maps and associated DTDs into the XML Gateway repository. The XML Gateway Execution Engine and the XML Parser use the maps to create outbound messages and to consume inbound XML messages.

## Message Designer Menus

The Message Designer has the following menus: File, View, and Help. Some of the menu functions can also be accessed by toolbar, page 2-3 icons.

See: Message Designer Toolbar, page 2-3.

### File Menu

Use the Message Designer to create new data definitions and transaction maps or to modify existing data definitions and transaction maps.

The Message Designer supports the following file types:

- XGD for source or target data definitions
- XGM for transaction maps

The .XGD and .XGM files are XML files that can be opened and read using a browser or any XML editor. The XGD and XGM files are used to describe the structure and content of a message. The XML message produced or consumed using the transaction map (XGM) contains the actual business data.

The File menu options are listed in the following table:

Menu Option	Description
<b>New</b>	Create a new data definition file or a new transaction map file.
<b>Open</b>	Open an existing file. Files can be either data definition files with a XGD extension, or transaction map files with a XGM extension.
<b>Close</b>	Close the open transaction map or data definition file.
<b>Save</b>	Save an open file. Save the data definition files as XGD files. Save the transaction map files as XGM files.
<b>Properties</b>	Provides access to key property values entered using the Data Definition Creation Wizard or Map Creation Wizard. Use this menu option to change any of the original key values presented. See File > Properties Menu Option, page 2-4 for more details regarding this menu option.
<b>Exit</b>	Exit Message Designer.

### View Menu

Use the View Menu to view the source or target definitions in tree format only, table format only, or both tree and table formats.



The View menu options are listed in the following table:

Menu Option	Description
<b>Tree</b>	View source or target definition in tree format only.
<b>Table</b>	View source or target definition in table format only.
<b>View Both</b>	View source or target definition in both tree and table formats. This is the default viewing option.

## Help Menu

The Help menu options are listed in the following table:

Menu Option	Description
<b>Help Topics</b>	Displays Message Designer help.
<b>About</b>	Displays the Message Designer version.

## Message Designer Toolbar

The Message Designer toolbar uses the following icons to duplicate the noted menu options:



1. **Create New Map** icon. Invokes the Map Creation Wizard.
2. **Open Map** icon. Opens an existing map.
3. **Save Map** Icon. Saves the data definition as a XGD file. Saves the transaction map as a XGM file.
4. **View Tree** icon. Displays the tree format only.
5. **View Table** icon. Displays the table format only.
6. **View Both Tree and Table icon.** Displays both the tree and table formats.
7. **Help** icon. Invokes Message Designer Help.

## Message Designer Buttons



The **Add Sibling Button** adds a new element at the same hierarchy level as the selected item on the map.

For additional information on this function, see the following:

For Source Data Definitions see (Source Definition Tab) Add Sibling, page 2-37.

For Target Data Definitions see (Target Definition Tab) Add Sibling, page 2-43.



The **Add Child** button adds a new element at a lower hierarchy level than the selected item on the map.

For additional information on this function, see the following:

For Source Data Definitions see (Source Definition Tab) Add Child, page 2-37.

For Target Data Definitions see (Target Definition Tab) Add Child, page 2-43.



The **Delete** button deletes any element that has not been mapped. Be careful not to delete required data elements. Deleting required elements will cause a parser violation.

If an item has child elements associated with it, a warning is displayed before the delete occurs.

**Important:** If you are deleting any DTD extensions, be sure to remove the DTD extensions from the corresponding extension file created for the application or user. The extra information will not cause a parser violation, but it is a good practice to ensure the extension files match the message maps. Refer to How to Extend DTDs, page 2-84 for the details.

## File > Properties Menu

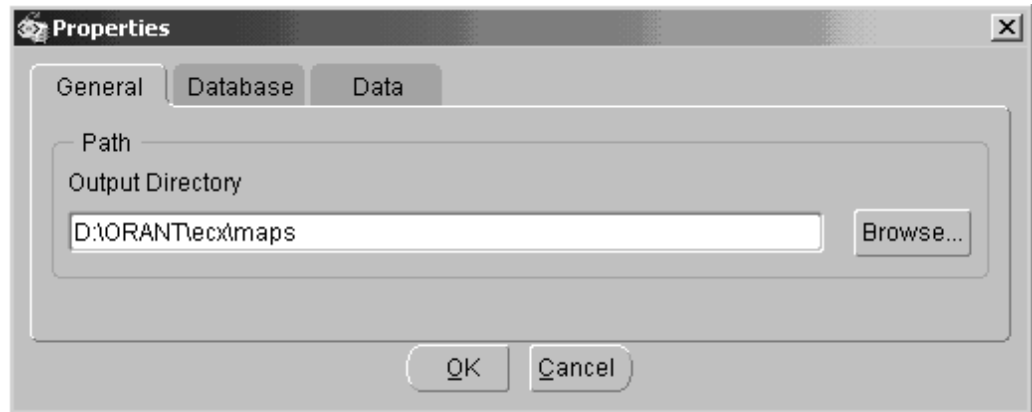
The File > Properties menu option provides access to key property values entered using the Data Definition Creation Wizard or the Map Creation Wizard. Use the Properties window to change any of the key property values presented.

The Property Tabs and fields associated with each Tab vary depending on where the File > Properties menu option is invoked. The options available from each window are:

- Main Message Designer - General Tab, page 2-4, Database Tab, page 2-5
- Data Definition Window - General Tab, page 2-4, Database Tab, page 2-5, Data Tab, page 2-6
- Transaction Map Window- General Tab, page 2-4, Database Tab, page 2-5, Map Tab, page 2-7, Source Tab, page 2-8, Target Tab, page 2-9

## General Tab

The General Tab allows you to update the Output Directory. This is the default directory used to store the data definition and message map files created using the Message Designer.



### Output Directory

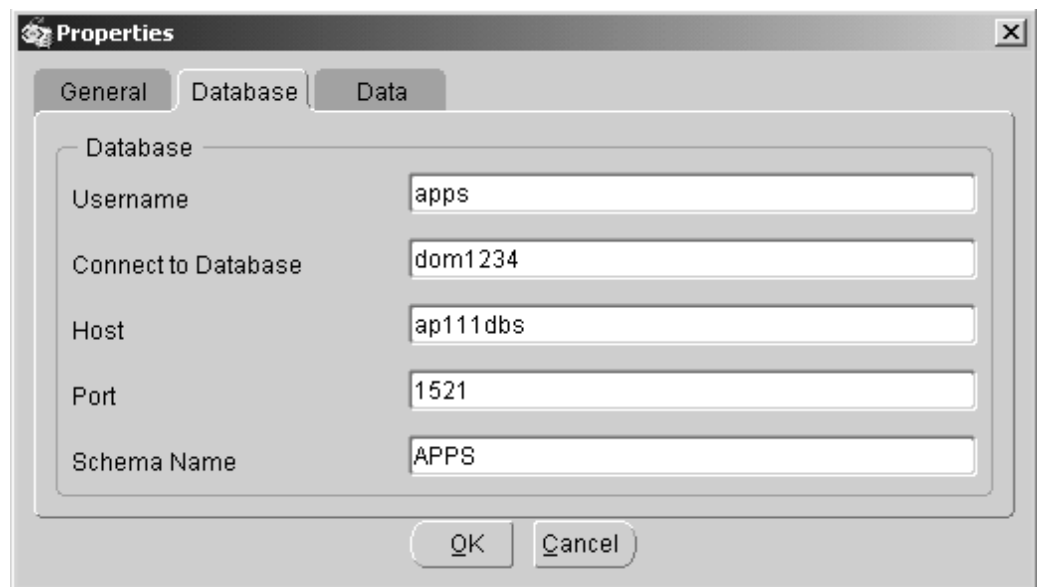
Use the Browse button to select a default directory or enter a valid directory name.

### Database Tab

Use the Database Tab to provide the default database connection information. The default values will be provided to the Data Definition and Map Creation Wizards as well as to the Procedure Call action.

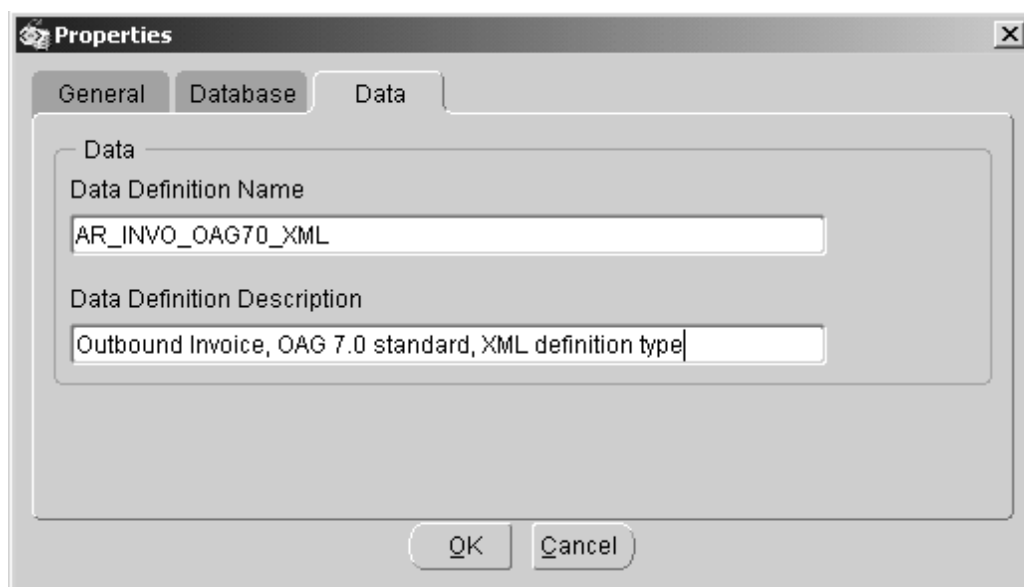
The database connection fields prompted for are:

- Username
- Connect to Database
- Host
- Port
- Schema Name



## Data Tab

The Data Tab allows you to update data values originally entered using the Data Definition Creation Wizard. The fields on the tab vary depending on the type of data definition.



The screenshot shows a 'Properties' dialog box with three tabs: 'General', 'Database', and 'Data'. The 'Data' tab is active. Inside the 'Data' tab, there is a section titled 'Data' containing two text input fields. The first field, 'Data Definition Name', contains the text 'AR\_INVO\_OAG70\_XML'. The second field, 'Data Definition Description', contains the text 'Outbound Invoice, OAG 7.0 standard, XML definition type'. At the bottom of the dialog are 'OK' and 'Cancel' buttons.

The fields in the following table display for all data definitions:

Field	Description
Data Definition Name	Update the name entered. Observe the naming conventions recommended in the Data Definition and Map Creation Wizards. See Select/Create a Source/Target Data Definition Name and Type, page 2-20 for the naming convention details.
Data Definition Description	Update the description.

**Important:** If the data definition changes affect any existing maps, the maps must be changed and reloaded. If the DTD reference is changed, the new DTD must be reloaded. Refer to How to Load/Delete Message Maps and DTDs, page 2-87.

The fields in the following table display when the data definition type is XML:

Field	Description
Root Element	Update the Root Element entered. The root element entered must match the root element of the DTD entered below.
Runtime DTD Location	Update the Runtime DTD Location with the new subdirectory name. Observe the naming conventions recommended in the Data Definition and Map Creation Wizards. See Identify the Runtime Location of a DTD, page 2-31 for naming convention details.
DTD File Name	Use the Browse button to select a DTD or enter a valid DTD file name. This will replace the file name originally entered. The root element defined in the DTD must match the root element value entered above.

**Important:** Changes to any of these property values require the message map to be reloaded. Changing the DTD associated with the message map requires the DTD to be reloaded. Refer to How to Load/Delete Message Maps and DTDs, page 2-87 for details.

## Map Tab

This tab displays only if you are viewing a map transaction file (.XGM).

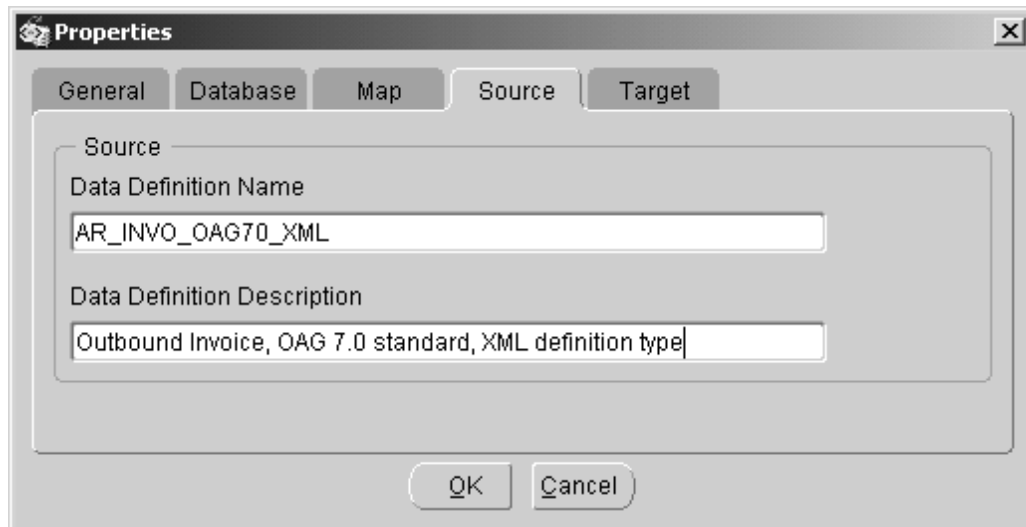
The screenshot shows a 'Properties' dialog box with the 'Map' tab selected. The 'Map' section contains a 'Map Name' text field and a 'Map Description' text area, both containing the text 'AR\_PROCESS\_INVOICE\_OAG\_OUT'. The dialog also has 'General', 'Database', 'Source', and 'Target' tabs, and 'OK' and 'Cancel' buttons at the bottom.

Field	Description
Map Name	Update the name entered. Observe the naming conventions recommended in the Map Creation Wizard. See Specify a Map Name, page 2-19 for details.
Map Description	Update the description.

**Important:** If the map name change affects any existing Trading Partner definitions, update the Trading Partner definition with the new map name.

**Important:** The map containing the new name must be reloaded into the XML Gateway repository. Refer to How to Load/Delete Message Maps and DTDs, page 2-87 for the details.

## Source Tab



The fields in the following table display for all data definitions:

Field	Description
Data Definition Name	Update the name entered. Observe the naming conventions recommended in the Data Definition and Map Creation Wizards. See Specify Source/Target Data Definition Name and Type, page 2-21 for naming convention details.
Data Definition Description	Update the description.

**Important:** If the data definition changes affect any existing maps, the maps must be changed and reloaded. If the DTD reference is changed, the new DTD must be reloaded. Refer to How to Load/Delete Message Maps and DTDs, page 2-87.

The fields in the following table display when the data definition type is XML:

Field	Description
Root Element	Update the Root Element entered. The root element entered must match the root element of the DTD entered below.
Runtime DTD Location	Update the Runtime DTD Location with the new subdirectory name. Observe the naming conventions recommended in the Data Definition and Map Creation Wizards. See Identify the Runtime Location of a DTD, page 2-31 for naming convention details.
DTD File Name	Use the Browse button to select a DTD or enter a valid DTD file name. This will replace the file name originally entered. The root element defined in the DTD must match the root element value entered above.

**Important:** Changes to any of these property values require the message map to be reloaded. Changing the DTD associated with the message map requires the DTD to be reloaded. Refer to How to Load/Delete Message Maps and DTDs, page 2-87 for details.

## Target Tab

The screenshot shows the 'Properties' dialog box with the 'Target' tab selected. The dialog contains the following fields and values:

- Data Definition Name:** AR\_INVO\_OAG70\_XML
- Data Definition Description:** Outbound Invoice, OAG 7.0 standard, XML definition type
- Root Element:** PROCESS\_INVOICE\_001
- Runtime DTD Location:** AR
- DTD File Name:** 901\_process\_invoice\_001.dtd (with a 'Browse...' button next to it)

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

The fields in the following table display for all data definitions:

Field	Description
Data Definition Name	Update the name entered. Observe the naming conventions recommended in the Data Definition and Map Creation Wizards. See Specify Source/Target Data Definition Name and Type, page 2-21 for naming convention details.
Data Definition Description	Update the description.

**Important:** If the data definition changes affect any existing maps, the maps must be changed and reloaded. If the DTD reference is changed, the new DTD must be reloaded. Refer to How to Load/Delete Message Maps and DTDs, page 2-87.

The fields in the following table display when the data definition type is XML:

Field	Description
Root Element	Update the Root Element entered. The root element entered must match the root element of the DTD entered below.
Runtime DTD Location	Update the Runtime DTD Location with the new subdirectory name. Observe the naming conventions recommended in the Data Definition and Map Creation Wizards. See Identify the Runtime Location of a DTD, page 2-31 for naming convention details.
DTD File Name	Use the Browse button to select a DTD or enter a valid DTD file name. This will replace the file name originally entered. The root element defined in the DTD must match the root element value entered above.

**Important:** Changes to any of these property values require the message map to be reloaded. Changing the DTD associated with the message map requires the DTD to be reloaded. Refer to How to Load/Delete Message Maps and DTDs, page 2-87 for details.

## Message Designer Wizards

The Message Designer contains two wizards to guide you through the map creation process. The two wizards are:

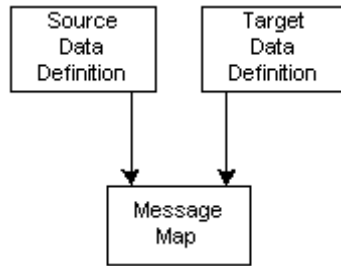
- Data Definition Creation Wizard
- Map Creation Wizard

The Data Definition Creation Wizard is used to define the data definitions.

A data definition is a collection of data used to describe a business object such as a customer profile or an invoice document. A data definition may be based on database views, database tables, application open interface tables, an application API, an XML Document Type Definition (DTD), or a production XML message.

A message map consists of a source data definition and a target data definition graphically represented as follows:





The data definitions created using the Data Definition Creation Wizard are used to create a preliminary message map. The Data Definition Creation Wizard is ideal for creating a master definition to use as the basis for creating trading partner-specific message maps.

Three combinations of source and target data definitions are supported by the Message Designer as described in the following table:

Source	Target	Purpose
Database	XML	Outbound XML message
XML	Database	Inbound XML message
XML	XML	Transform one version of a DTD to the next version of the same DTD.  Transform from DTD in one standard to DTD of another standard as long as the DTDs are for the same business function.  Pass-through XML message (Refer to How to Map a Pass-Through Message, page 2-86)  Mapping to an Application API (Refer to How to Map to an API, page 2-86)

Database-based data definitions represent a description of the Oracle data model required to support the XML message.

XML-based data definitions are based on an XML Document Type Definition (DTD) or a production XML message. XML to XML transformations from one DTD to another DTD where the DTDs are for different business purposes is done in two steps:

1. Database to XML
2. XML to database

The Map Creation Wizard is used to define the following:

- Source Definition
- Target Definition
- Preliminary Message Map

The source and target data definitions may be new or based on definitions previously created using the Data Definition Wizard. The source and target data definitions form the basis of a preliminary message map.

The Message Designer is used to define the following to complete the message map:

- Level Mapping

- Element Mapping
- Actions

Level mapping is the process of relating the source data structure to the target data structure. Element mapping is the process of relating a source data element to a target data element. Actions are data transformation or process control functions that can be applied at the data element, document, or root level.

## Data Definition Creation Wizard Process Flow

Creating a Data Definition consists of two general steps:

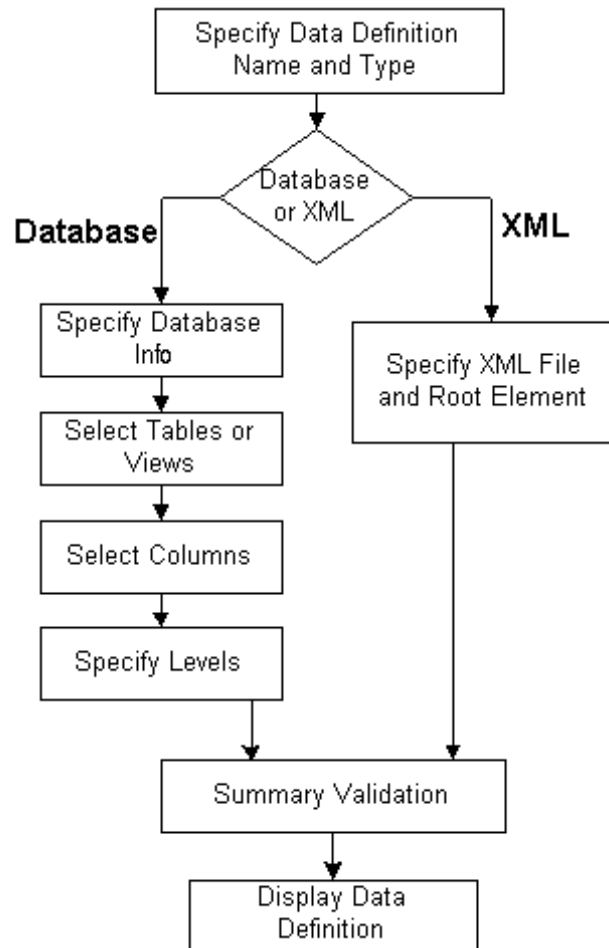
Step 1: Name the data definition and select the type

Step 2: Identify data definition details

The tasks required to complete Step 2 vary depending on your selections in Step 1.

The illustration below displays the flow of the screens presented to you in the Data Definition Creation Wizard. After you specify the data definition name and type (Step 1), you are prompted to provide the details based on whether you chose a type of database or XML.

When you have completed Step 2, you are presented with a summary screen where you can either finish or return to previous steps to make edits. Clicking Finish will close the Wizard and open the Message Designer Data Definition window.



## Map Creation Wizard Process Flow

There are three general steps to creating a map. Each step consists of a variable number of tasks depending on the selections you make in the Wizard:

Step 1: Specify a Map Name

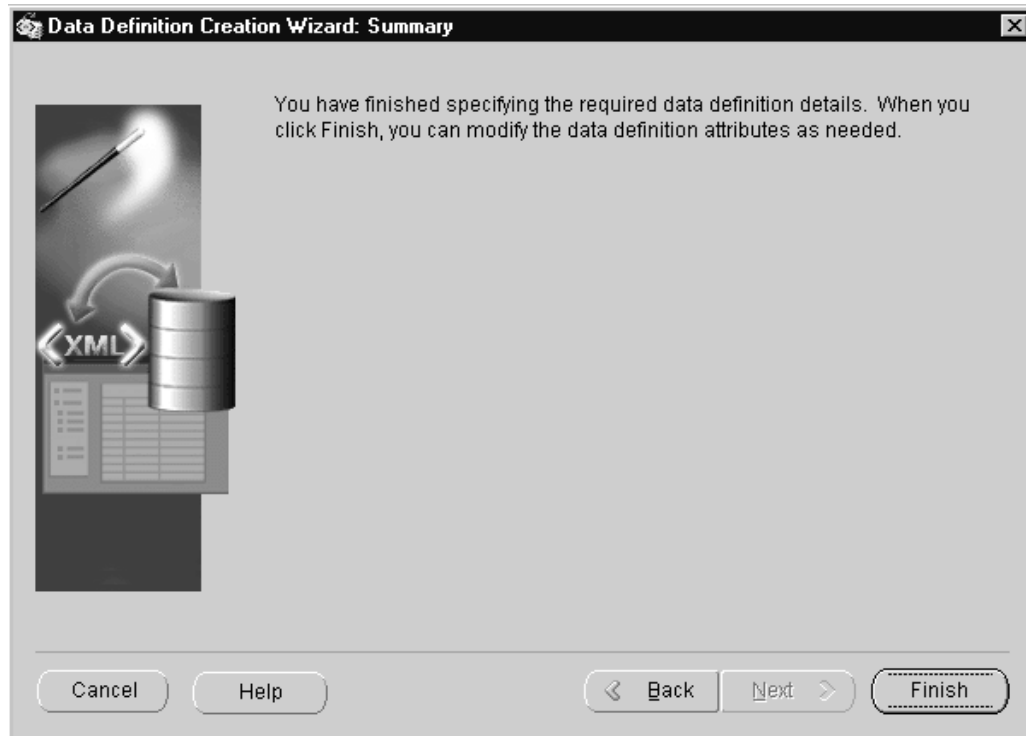
Step 2: Select/Create a Source Data Definition

Step 3: Select/Create a Target Data Definition

The illustration below displays the flow of the screens presented to you in the Map Creation Wizard. After you specify a map name (Step 1), you are prompted to select or create a source data definition (Step 2). If you choose to **create** the data definition, you are guided through the data definition creation steps (identical to the Data Definition Creation Wizard steps).

After you create your source data definition, or if you chose to **select** an existing data definition, you are prompted to select or create a target data definition (Step 3). The target data definition steps are identical to the source data definition steps.

When you have completed Step 3, you are taken to a summary screen where you can either finish or return to previous steps to make edits.



## Using the Data Definition Creation Wizard

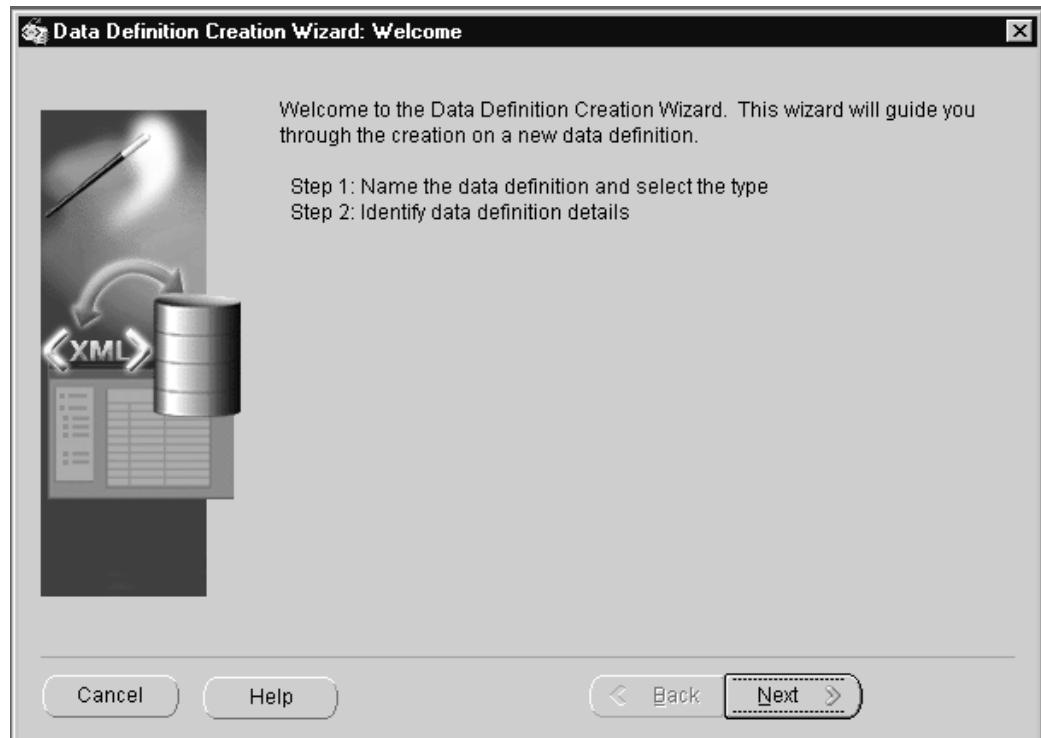
The Data Definition Creation Wizard guides you through the data definition creation process. The data definition is saved as a .xgd file and can be selected as a target or a source when you create a map.

**Important:** It is not necessary to create your data definitions before you begin the map creation process. The Map Creation Wizard gives you the option of choosing an existing data definition or creating a new one. If you choose to create a new one, you will be guided through the data definition creation steps as part of your map creation process. See Map Creation Wizard, page 2-17.

Launch the Data Definition Creation Wizard by selecting

**(M) File > New > Data Definition**

The first window welcomes you to the wizard.



The following buttons are available for all Wizard steps:

- **Cancel**  
Exit the Wizard and cancel all completed Wizard steps.
- **Back**  
Return to the previous Wizard step.
- **Next**  
Advance to the next Wizard step. The Next button is enabled only after the required information for the current Wizard step has been entered.

## Data Definition Creation Wizard Steps

Follow the instructions presented for the Map Creation Wizard to complete each of the Data Definition Creation Wizard steps:

### Specify Data Definition Name and Type

See Specify Source/Target Data Definition Name and Type, page 2-21.

#### If you selected Database...

Complete the following steps:

##### **Specify Database Information**

See Specify Source/Target Definition Database Information, page 2-23.

##### **Select Tables or Views**

See Select Source/Target Tables or Views, page 2-24

**Select Columns**

See Select Source/Target Columns, page 2-26

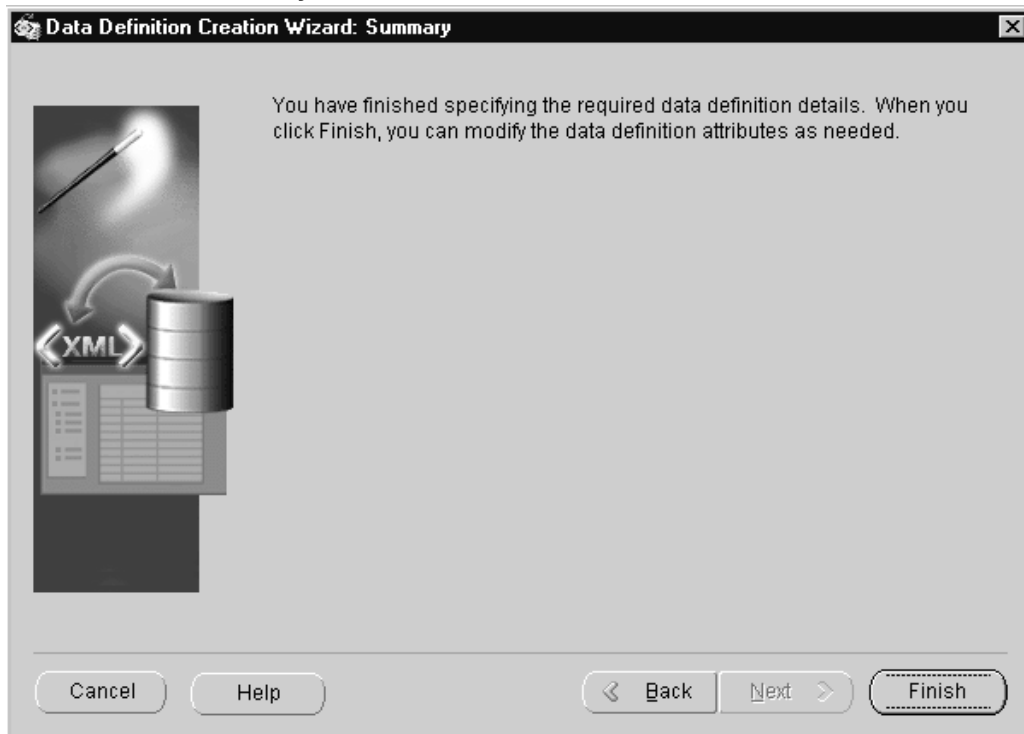
**Specify Source/Target Levels**

See Specify Source/Target Levels, page 2-28.

**If you selected XML...**

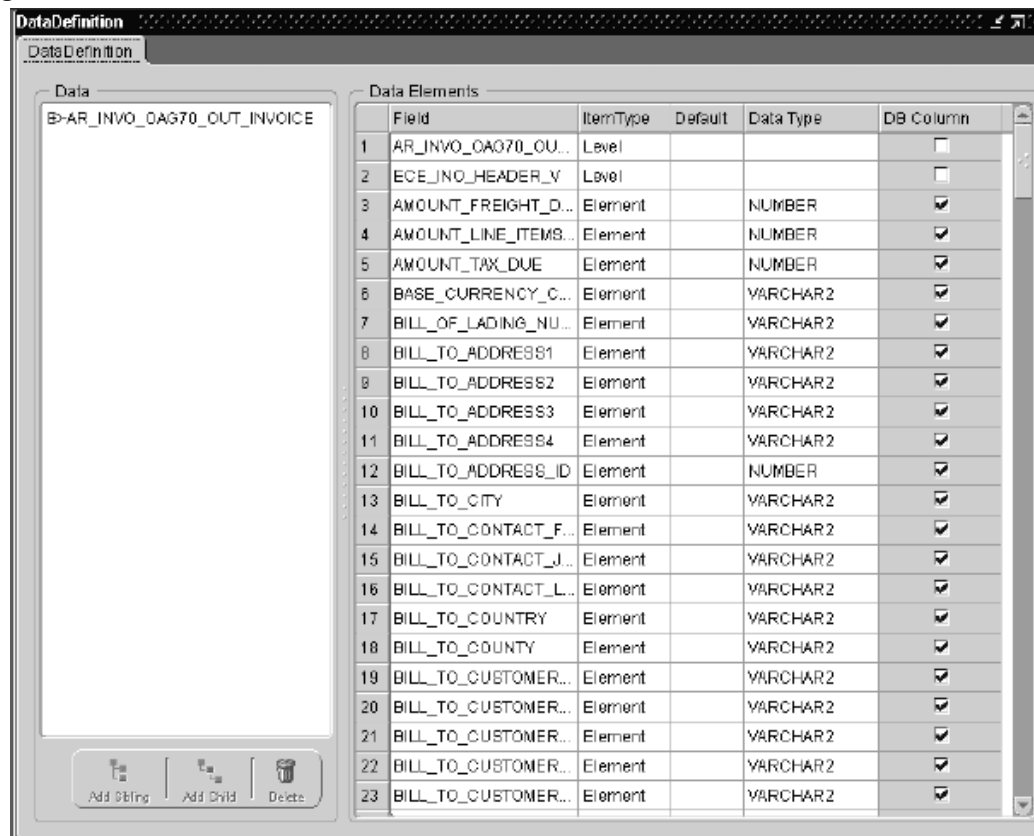
If you selected XML as the data definition type, you will be prompted for the XML information.

See Specify Source/Target XML File and Root Element, page 2-30.

**Data Definition Creation Wizard: Summary**

Click Finish to complete the Wizard steps. If you wish to change any selections you have made in defining your data definition, you can use the Back button to return to the appropriate Wizard window.

## Message Designer Data Definition Window



The data definition you just created is displayed in the Data Definition tab. You can extend the definition using the Data Definition window to perform the following:

- Provide default values
- Create additional nodes and elements using the Add Sibling button
- Add fields using the Add Child button
- Enable code conversion
- Define conditional node mapping rules for additional nodes or element (if source is DTD)

Once the data definition is complete, use the File > Save (Data Definition) menu option or the Toolbar equivalent. Use the File > Properties menu to change the default directory if necessary.

## Using the Map Creation Wizard

The Map Creation Wizard guides you through the map creation process. A series of windows are presented to create a map based on the source and target definitions.

Launch the Map Creation Wizard by selecting

**(M) File > New > Transaction Map**

or by clicking the **Create New Map** icon on the toolbar. The first window welcomes you to the wizard.



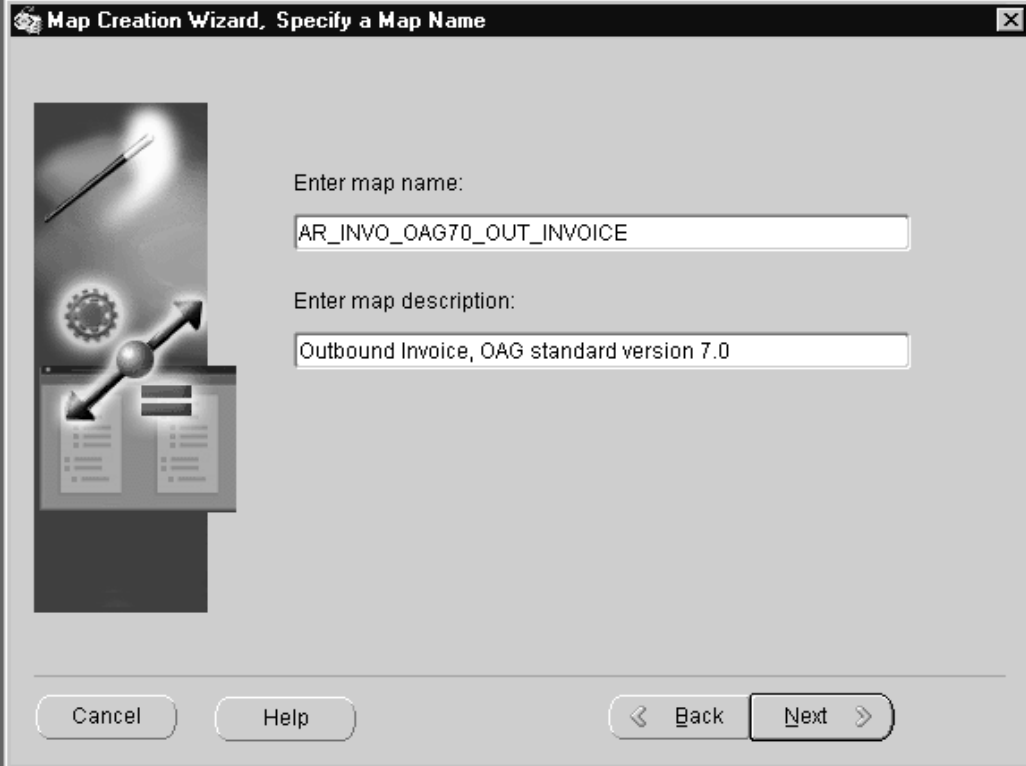
The following buttons are available for all Wizard steps:

- **Cancel**  
Exit the Wizard and cancel all completed Wizard steps.
- **Back**  
Return to the previous Wizard step.
- **Next**  
Advance to the next Wizard step. The Next button is disabled until you supply all required data for a step.

Click Next to continue. The Wizard will display the Specify a Map Name window.



## Specify a Map Name



Map Creation Wizard, Specify a Map Name

Enter map name:  
AR\_INVO\_OAG70\_OUT\_INVOICE

Enter map description:  
Outbound Invoice, OAG standard version 7.0

Cancel Help < Back Next >

Specify a unique map name. This is the name assigned to the message map you are creating. It is also the name to be used when associating a message map to a Trading Partner.

## Enter Map Name

Enter a unique map name. This name is stored as the map code in the map definition file.

In addition to being unique, the name should describe the intended use of the map for easy identification. The recommended naming convention is as follows:

- Product mnemonic or user ID
- Transaction subtype as entered in the Define Transactions form
- XML standard and version used (for example, OAG, Rosettanet, iFX)
- Inbound or outbound message

Examples of map names using the recommended naming convention are:

- AR\_INVO\_OAG70\_OUT\_INVOICE

This map name is used for an Oracle Receivables outbound invoice message that uses the OAG standard, version 7.0 DTD.

- USER\_ACK\_OAG70\_IN

This name represents a user-developed map for an inbound acknowledgment message that uses the OAG standard, version 7.0 DTD.

**Note:** You can use the File > Properties menu option to change the Map Name and Map Description as necessary.

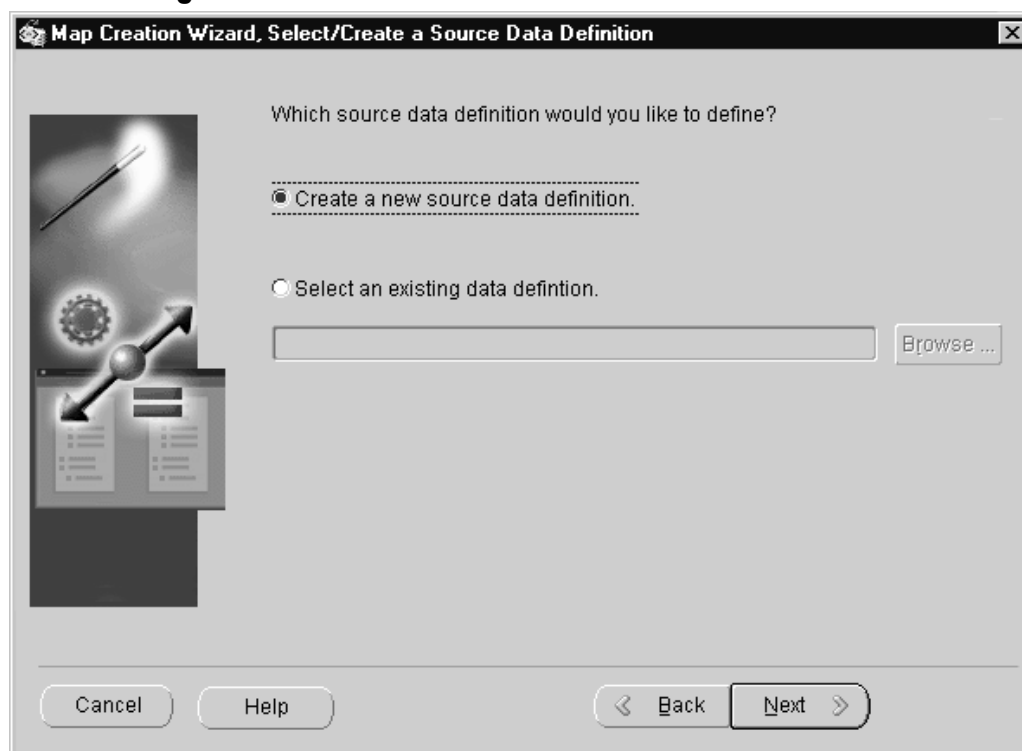
## Enter Map Description

Enter a description for the map.

**Note:** You can use the File > Properties menu option to change the Map Name and Map Description as necessary.

Click Next to continue. The Wizard will display the Select/Create a Source Data Definition window.

## Select/Create a Source/Target Data Definition



Select an existing data definition file or create a new data definition file.

## Create a new source/target data definition

Select this option to create a new data definition. The Wizard will guide you through creating your new .XGD file.

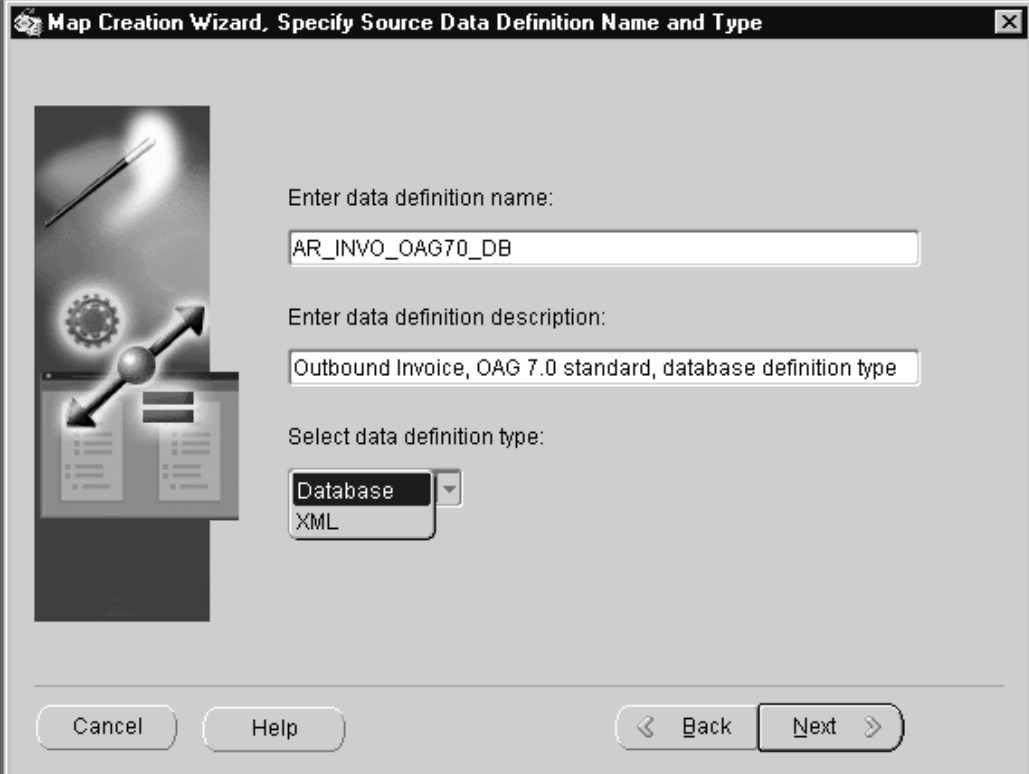
## Select an existing data definition

Existing definition files are stored on the file system as *<filename>.XGD*. Use the Browse button to view the available data definition files.

## Click Next to continue

- If you are defining your **Source data definition** and you selected...
  - **Create** a new source data definition, proceed to Specify Source/Target Data Definition Name and Type, page 2-21.
  - **Select** an existing Source data definition, repeat this step for your Target Data Definition.
- If you are defining your **Target data definition** and you selected...
  - **Create** a new target data definition, proceed to Specify Source/Target Data Definition Name and Type, page 2-21.
  - **Select** an existing Target data definition, proceed to Map Creation Wizard Summary, page 2-32.

## Specify Source/Target Data Definition Name and Type



Map Creation Wizard, Specify Source Data Definition Name and Type

Enter data definition name:  
AR\_INVO\_OAG70\_DB

Enter data definition description:  
Outbound Invoice, OAG 7.0 standard, database definition type

Select data definition type:  
Database  
XML

Cancel Help < Back Next >

For a new data definition, enter a data definition name, description, and type.

## Enter Data Definition Name

Enter a name that describes the contents of the data definition to allow easy identification.

This name is displayed in the Transaction Map window as the root node, if you select database as the data definition type. If you select XML as the data definition type, the DTD root element is displayed in the Transaction Map window as the root node instead of the data definition name entered here.

The recommended naming convention is as follows:

- Product mnemonic or User ID
- Transaction subtype as entered in the Define Transactions form
- XML standard and version used (for example, OAG, Rosettanet, iFX)
- Database or XML data definition type

Examples of data definition names using the recommended naming convention are:

- AR\_INVO\_OAG70\_DB

This name describes an Oracle Receivables-developed outbound invoice message that uses the OAG standard, version 7.0 DTD. This data definition is for the Database data definition type intended for use as the source data definition for an outbound message.

- USER\_ACK\_OAG70\_XML

This name describes a user-developed inbound acknowledgement message that uses the OAG standard, version 7.0 DTD. This data definition is for the XML data definition type intended for use as the source data definition for an inbound message.

Use the File > Properties menu option to change the Data Definition Name as necessary.

## Enter Data Definition Description

Enter a description for the data definition.

Use the File > Properties menu option to change the Data Definition Description as necessary.

## Select Data Definition Type

Select a data definition type from the following list of values:

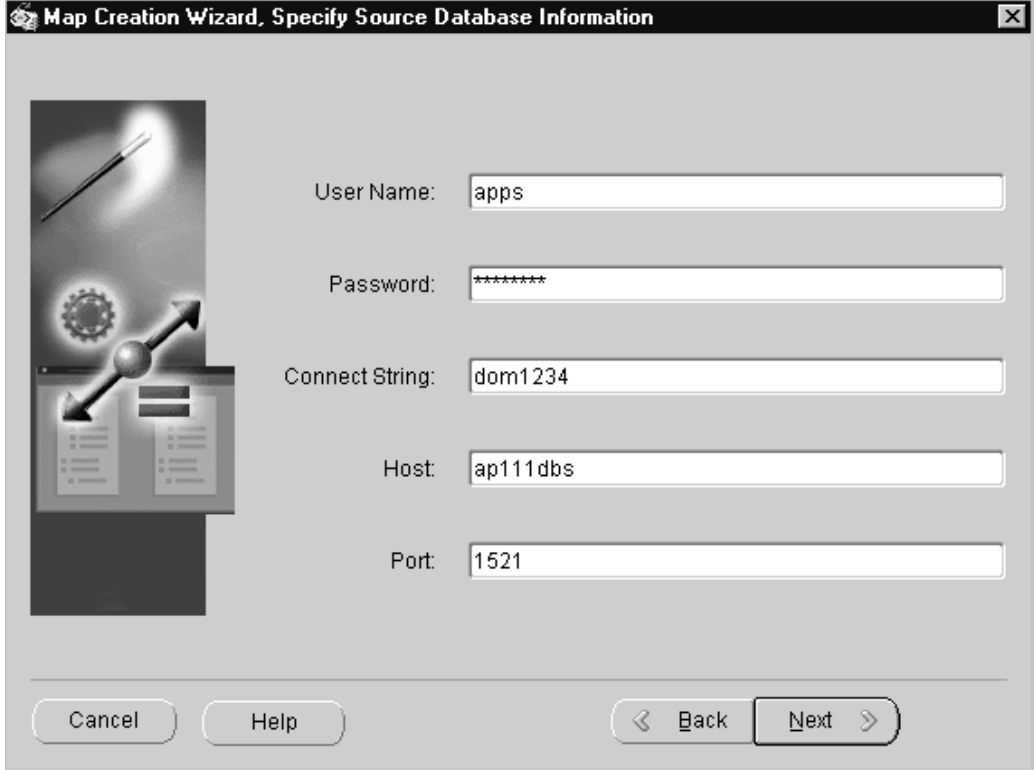
- Database
- XML

## Click Next to continue

The wizard steps to follow will vary based on the data definition type selected.

- If you selected **Database**, proceed to Specify Source/Target Definition Database Information, page 2-23.
- If you selected **XML**, proceed to Specify Source/Target XML File and Root Element, page 2-30.

## Specify Source/Target Definition Database Information



The dialog box is titled "Map Creation Wizard, Specify Source Database Information". It features a vertical toolbar on the left with icons for a pencil, a gear, a double-headed arrow, and a list. The main area contains five text input fields, each with a label to its left: "User Name:" with the value "apps", "Password:" with "\*\*\*\*\*", "Connect String:" with "dom1234", "Host:" with "ap111dbs", and "Port:" with "1521". At the bottom, there are four buttons: "Cancel", "Help", "< Back", and "Next >".

The default database access information is displayed from the Database tab, page 2-5 of the File > Properties menu. Enter the password and make changes to the database access information if necessary.

**Note:** Changes made on this screen are not copied back to the Properties file. Entries on this screen are used for the current session only.

This step does not apply if the source/target is a DTD because database access is not required.

### User Name

Enter the user name for the database schema to be accessed.

### Password

Enter the password for your User Name.

### Connect String

Enter the connect string for the database.

### Host

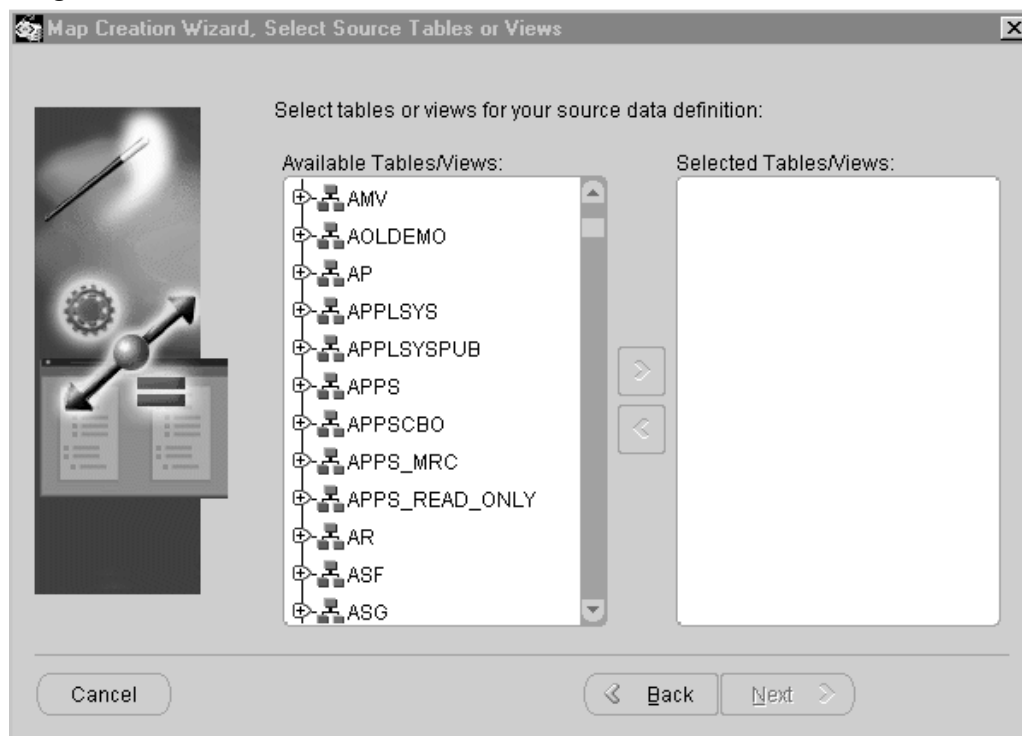
Enter the host name for the database.

## Port

The default value of "1521" is displayed. Enter another valid port value (if necessary) for the database to be accessed.

Click Next to continue. The Wizard will display the Select Source/Target Tables or Views window.

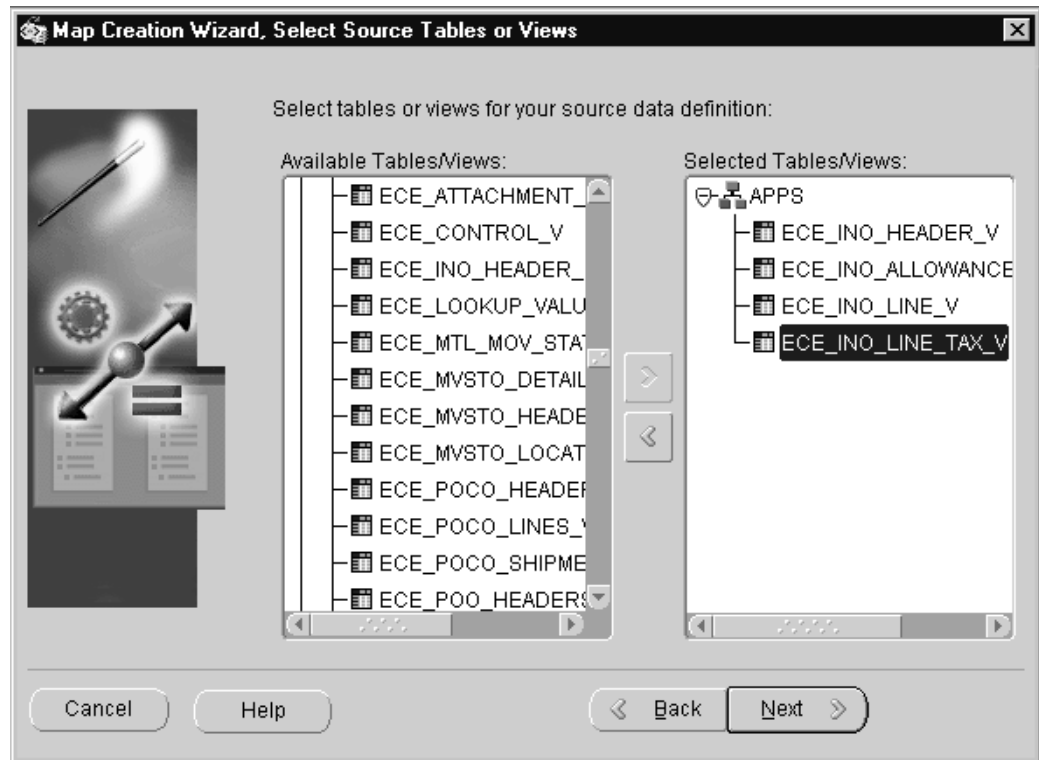
## Select Source/Target Tables or Views



If you selected a Data Definition Type of Database, you will be prompted for the database schema, database views, and tables required by the message.

All application database views are defined in the APPS schema. The associated database tables are defined in the application-specific database schema. Access to application-specific database tables must be granted by defining a synonym in the APPS schema.

For applications with database views or tables shared across multiple hierarchical levels of the data model representing the business document, a database view or table alias is required for the second and each subsequent use of the view or table. This is necessary because this wizard step deletes a database view or table from the Available Tables/Views list once it has been selected.



This step does not apply if the source/target is a DTD, because database information is not necessary.

## Available Tables/Views

Expand the APPS database schema tree to view all the available database tables and views.

## Selected Tables/Views

Select the desired database views and tables from the left window, click on the right shuttle to move the selected database view or table from Available Tables/Views to Selected Tables/Views. Continue this process until you have selected all the required database views and tables for each schema selected.

Once you have selected all the database views and tables required by the message, select any special XML Gateway database views you may have defined to support the specific requirements of the XML standard you are using.

For example, select the ECX\_OAG\_CONTROLAREA\_TP\_V (formerly ECX\_OAG\_CONTROLAREA\_V) database view to map to the OAG CNTROLAREA data type.

**Note:** The ECX\_OAG\_CONTROLAREA\_TP\_V view is an upgraded version of the ECX\_OAG\_CONTROLAREA\_V view. Oracle XML Gateway supports both versions of the database view.

The upgraded view includes new fields for USERNAME, SOURCE\_TP\_LOCATION\_CODE, PARTY\_ID, PARTY\_SITE\_ID, and PARTY\_TYPE as well as changes to the following existing fields:

**REFERENCE\_ID** is based on the system name, event name, and event key defined by the application business event. **REFERENCE\_ID** was previously defaulted to "1" with recommendations to use the **ECX\_REFERENCE\_ID** sequence to get a unique number. The use of this field varies by message map.

You must add the **ECX\_EVENT\_MESSAGE** item attribute to your Workflow item type to have access to the event details.

**CONFIRMATION** is based on the setting defined for the Trading Partner and business document entered using the Define Trading Partners window. **CONFIRMATION** was previously defaulted to "0".

**COMPONENT** is based on the internal transaction type entered on the Define Transactions window for the business document. **COMPONENT** was previously based on the external transaction type.

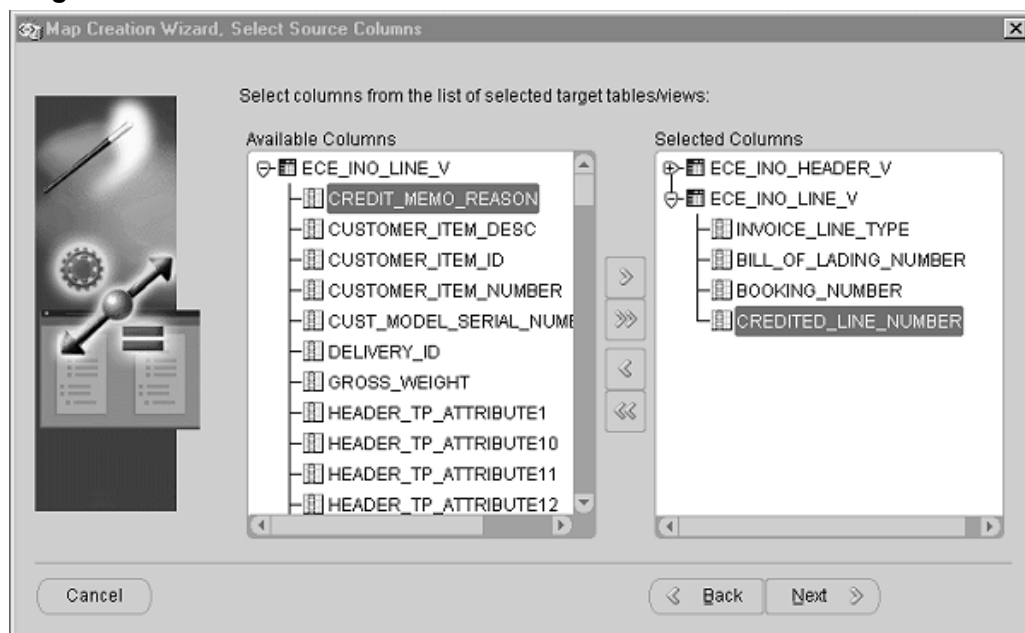
**TASK** is based on the internal transaction subtype entered on the Define Transactions window for the business document. **TASK** was previously based on the external subtype.

**TRANSACTION\_SUBTYPE** is based on the internal transaction subtype entered on the Define Transactions window. **TRANSACTION\_SUBTYPE** was previously defaulted to the **TRANSACTION\_TYPE** if no value was found in the database.

To deselect a selected database view or table, select the desired database view/table from the right window, click on the left shuttle button to move the selection from the Selected Tables/Views back to Available Tables/Views.

Click Next to continue. The Wizard will display the Select Source/Target Columns window.

## Select Source/Target Columns





This window prompts you for the columns required from each of the database views or tables selected in the previous step.

This step does not apply if the source/target is a DTD because database information is not necessary.

## **Available Columns**

Expand each of the database views and tables to view all the available database view and table columns.

## **Selected Columns**

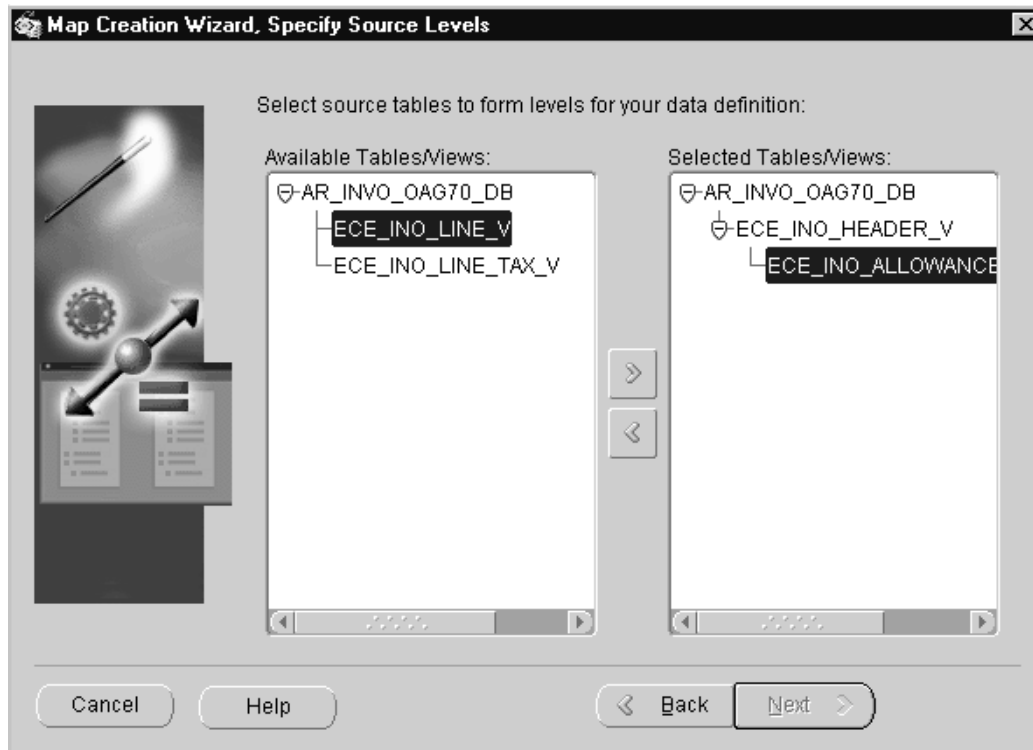
Select the desired database view or table column from the left window, click on the right shuttle button to move the selected database view or table column from Available Columns to Selected Columns. To move all the columns from a view or table, select the table or view and click the double right shuttle button to move all its columns from Available Columns to Selected Columns. Continue this process until all the required database view and table columns are selected.

You cannot proceed to the next wizard step until you have selected at least one column from each of the selected database views and tables. If necessary, return to the previous wizard step to deselect a database view or table and then resume the column selection process.

To deselect a selected database view or table column, highlight the desired column from the right window and click the left shuttle button to move the highlighted column from Selected Columns back to Available Columns. To deselect all the columns from a table or view, highlight the table or view name under Selected Columns and click the double left shuttle button to move all the columns back to the Available Columns area.

Click Next to continue. The Wizard will display the Specify Source/Target Levels window.

## Specify Source/Target Levels



If you selected a Data Definition Type of Database, you will be prompted to identify the hierarchy of the source or target data definition. This step is necessary only if your source/target data definition contains more than one level.

A level represents a collection of data that repeats. For example, Purchase Order lines represent a level within a Purchase Order because there are multiple PO lines to a single PO.

The purpose of this step is to identify the parent and child relationships of each database view. The source hierarchy will be used to relate to the target hierarchy as part of the map creation process.

This step does not apply if the source/target is a DTD because database information is not necessary.

### Available Tables/Views

The available database views and tables are displayed for your selection.

### Selected Tables/Views

This window allows you to define the parent and child relationships of the selected database views and tables along with the special database views necessary to relate the database data model to the DTD data model.

Start by identifying the parent node. For OAG, the parent node is the ECX\_OAG\_CONTROLAREA\_TP\_V view (formerly ECX\_OAG\_CONTROLAREA\_V). Related to the parent node are sibling or child nodes.

**Note:** The ECX\_OAG\_CONTROLAREA\_TP\_V view is an upgraded version of the ECX\_OAG\_CONTROLAREA\_V view. Oracle XML Gateway supports both versions of the database view. For more information see the Note, page 2-25.

Sibling and child relationships are defined by first identifying the parent node in the Selected Tables/Views window (right). The database view or table you move from the the Available window to the Selected window will always be added as the last child of the parent node selected.

Therefore, to specify a sibling relationship to a specific node, select its parent node in the Selected Tables/Views column (right window). Next select the database view or table in the Available Tables/Views (left window) that you want to define as its sibling. Click on the right shuttle. The selected database view or table will be displayed as the last child of the parent node selected, or as a sibling to the desired node.

To specify a child relationship to a specific node, select the node in the Selected Tables/Views column (right window). Next select the database view or table in Available Tables/Views (left window) that you want to define as its child, and click on the right shuttle. The selected database view or table will be displayed as the last child of the selected parent node.

Continue this process until all available database views and tables have been moved to Selected Tables/Views.

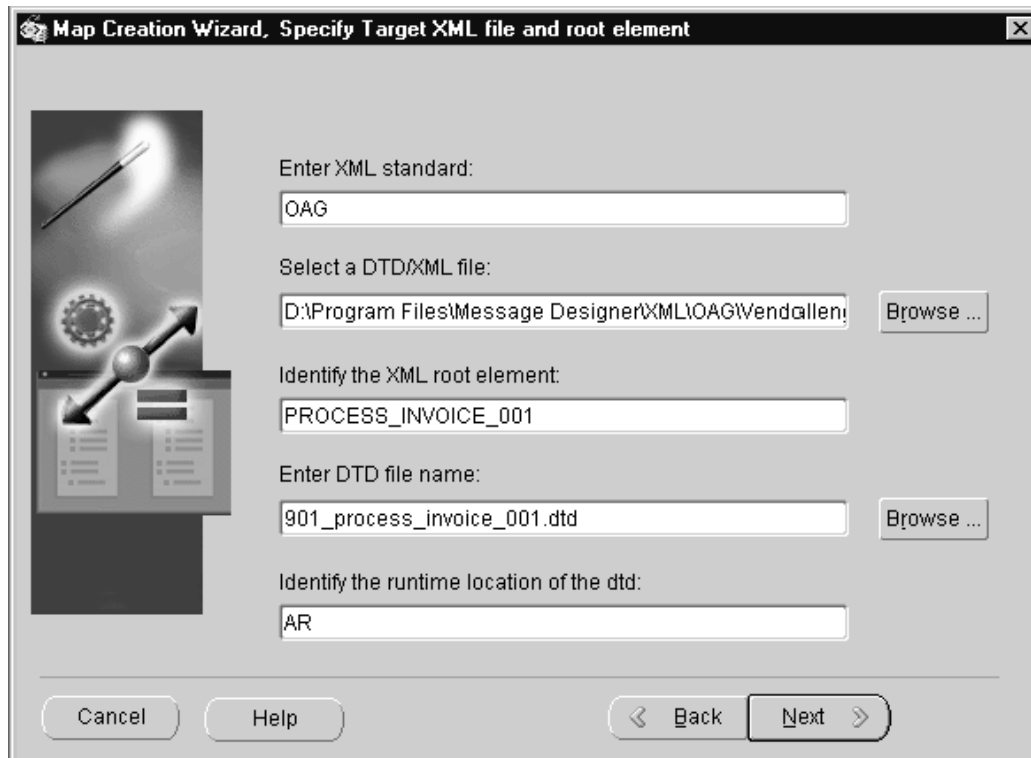
This completes the data definition process.

### **Click Next to continue**

If you have just finished your Source Data Definition, return to Select/Create a Source/Target Data Definition, page 2-20 to define your Target Data Definition.

If you have just finished your Target Data Definition, proceed to Map Creation Wizard Summary, page 2-32.

## Specify Source/Target XML File and Root Element



Map Creation Wizard, Specify Target XML file and root element

Enter XML standard:  
OAG

Select a DTD/XML file:  
D:\Program Files\Message Designer\XML\OAG\Vendor\... Browse ...

Identify the XML root element:  
PROCESS\_INVOICE\_001

Enter DTD file name:  
901\_process\_invoice\_001.dtd Browse ...

Identify the runtime location of the dtd:  
AR

Cancel Help < Back Next >

If you selected XML as the data definition type, you will be prompted for the XML information. You can use a DTD or a production XML message as the data definition.

### XML Standard

Enter the XML standard used.

"OAG" (Open Application Group XML Standard) is the default.

### Select a DTD/XML File

Select the DTD from the list of available DTDs presented when you click on the Browse button, or enter a specific DTD including the file path. The selected DTD and its file path are displayed.

To base the data definition on a production XML message, enter the file path and file name of the XML message.

The DTD or production XML message is used to create the definition tree for the Message Designer.

This step assumes that the required DTD and the associated external reference DTD files are available on the file system and are accessible by the Message Designer. For example, the OAG definition files are

- oagis\_domains.dtd
- oagis\_resources.dtd
- oagis\_fields.dtd

- oagis\_segments.dtd

## Identify the XML Root Element

Enter the XML root element. Open the DTD or production XML message using a browser or XML editor to determine the root element name if you do not know it.

**Note:** Because the representation of root element varies by XML standard, this step is required to inform the Execution Engine where to begin reading the DTD.

Use the File > Properties menu option to change the Root Element as necessary.

## Enter DTD File Name

If you entered a DTD above, the DTD file name will display automatically.

If you entered a production XML message above, click the Browse button to select the corresponding DTD from the list of available DTDs, or enter a specific DTD if available.

The DTD file name entered is used to validate a message to ensure that it is well-formed and valid before placing an outbound message on the outbound queue or processing a message dequeued from the inbound queue.

In addition, the primary DTD and its associated definition files must be loaded into the XML Gateway repository for access by the XML Gateway Execution Engine.

**Note:** Refer to How to Load/Delete Message Maps and DTDs, page 2-87 for more details.

Use the File > Properties menu option to change the DTD File Name as necessary.

## Identify the Runtime Location of a DTD

Enter the subdirectory name using the following naming convention:

`<application code>/xml/<standard><standards version>`

Examples:

`ar/xml/oag62`

`ap/xml/oag70`

**Note:** Do not use a period (.) when referencing a standards version.

The combination of the runtime DTD location and the DTD file name provides a unique identifier for the DTD required.

Use the File > Properties menu option to change the Runtime Location of a DTD as necessary.

## Click Next to continue

If you have just finished your Source Data Definition, return to Select/Create a Source/Target Data Definition, page 2-20 to define your Target Data Definition.

If you have just finished your Target Data Definition, proceed to Map Creation Wizard Summary, page 2-32.

## Map Creation Wizard Summary



Click Finish to exit the Map Creation Wizard. If you wish to change any selections you have made in defining your map, use the Back button to return to the appropriate Wizard window.

Before proceeding to the level and element mapping process, the XML Gateway, in conjunction with the XML Parser, performs the following validations:

- When the DOCTYPE tag is present in an XML message, verify that the DTD referenced is accurate.
- Verify that the XML root element matches the DTD identified. When a production XML message is identified, verify that the root element and DTD identified match the production XML message.
- Verify that external DTDs referenced by the primary DTD are available.
- Check for circular DTD references. Process the first occurrence, truncate the remainder and warn user to manually add the necessary repeating occurrences.

## Transaction Map Window

Upon exiting the Map Creation Wizard, the source and target definitions are presented in the Transaction Map window.

The Transaction Map window will also be presented if you select an existing map with a version number compatible with the version number of Message Designer.

The version number of the map (.xgm file) is stored in the <ECX\_MAJOR\_VERSION> and <ECX\_MINOR\_VERSION> tags. The Message Designer version number is available

in the Help > About menu. Maps are compatible with Message Designer if the major version is the same and the minor version is the same or lower.

The Transaction Map window is divided into four tabs as follows:

- Source Definition
- Target Definition
- Level Mapping
- Element Mapping (this tab appears only after a Level has been mapped)

## Source Definition

The source definition selected or created using the Map Creation Wizard is displayed in the Source Definition tab. You can extend the source definition to perform the following:

- Provide default values
- Enable code conversion
- Create duplicate nodes or elements using the Add Sibling button
- Add fields using the Add Child button
- Define conditional node mapping rules for duplicate nodes or elements (if the source is a DTD)

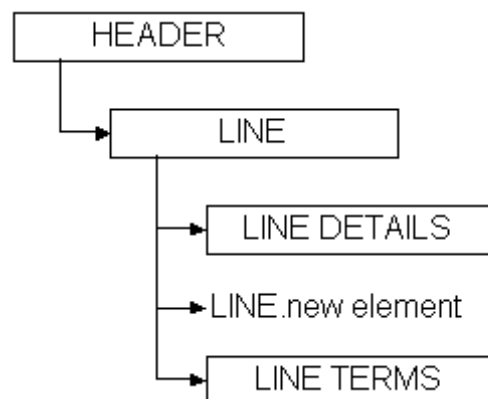
Once the source data definition is complete, it can be saved for reuse in other message maps. Use the File > Save (Data Definition) menu option or the Toolbar equivalent. Use the File > Properties menu to change the default directory or data definition property values if necessary.

If you do not wish to save the source data definition as an entity independent of the message map, then continue with the mapping process and save the transaction map at the end of the Element Mapping process.

## Source Definition Considerations

1. Refer to How to Map a Pass-Through Message, page 2-86, for guidelines related to developing a pass-through transaction.
2. Discontinuous nodes:

A discontinuous node is a non-level node that follows a level and is a sibling of that level. It can be represented graphically as follows:



In this example, the levels LINE DETAILS and LINE TERMS are children of the level LINE, and are siblings of each other. Inserted between LINE DETAILS and LINE TERMS is a new element of the LINE node. The new element is a continuation of the LINE node that creates a break between LINE DETAILS and LINE TERMS.

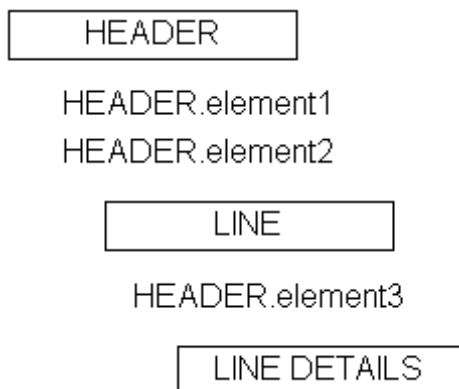
The ideal placement of the new element is as the last element of the LINE node before the LINE DETAILS node. However, some standards bodies do not have the flexibility to restructure an existing DTD, or may not wish to for backwards compatibility reasons. Regardless of the reason, the condition exists, and XML Gateway supports it.

For data definitions based on the Oracle data model, use the Add Child or Add Sibling button in the Message Designer, Transaction Map window to define a discontinuous node. Any new node introduced on the source and distributed across multiple target levels (expanded) or consolidated into a single target level (collapsed) will be grouped with the parent node and mapped according to the target definition. The rules against level cross-over still apply.

For data definitions based on a DTD, use the Transaction Map window, Item Type column to explicitly identify the data levels. This exercise may create a discontinuous node, which is not a problem unless you define the invalid scenario described below. The only user extensions supported for OAG DTDs are in the USERAREA. If you modify the DTD (using the Add Child or Add Sibling buttons) outside of the USERAREA, a parsing error will be triggered.

Message Designer will allow you to define a discontinuous node anywhere in the source or target definition. You will not know until runtime if the definition is valid or not. Therefore you should avoid introducing a discontinuous node for a node which also contains a child node.

The following graphic illustrates an invalid definition:



In the example above, HEADER.element3 is a continuation of the HEADER node. HEADER.element3 is also defined as a child of the LINE node. HEADER.element3 has a child node called LINE DETAILS. This is an invalid definition.

1. Refer to How to Implement the OAG Confirmation Business Object Document, page 4-13, for details on how to implement the optional confirmation message.
2. Refer to How to Implement Attachments in XML Messages, page 2-90 for details on how to include attachments with your XML documents.



## Source Definition Tab

### Field

Field identifies the name of the element, document, or root. The names are based on the database column names or DTD element names.

The field name may be changed if necessary, however, consider what this change implies. Because the field is based on the database column name or a DTD element name, corresponding changes may be necessary in the Oracle E-Business Suite or the DTD. The only changes allowed to a DTD are to the USERAREA. Refer to How to Extend DTDs, page 2-84 for details.

The first row is reserved for the Data Definition name (if the source is based on database views or tables) or root element name (if the source is a DTD).

Additional field names are displayed when sibling or child elements are added using the Add Sibling or Add Child buttons.

### Item Type

Item type identifies the field as either a Level or an Element. Level represents the parent in a parent-child relationship. Element represents the child in a parent-child relationship.

The Item Type of the first row is defaulted to "Level". You cannot change this value.

If the source is based on database views or tables, the Item Type for the database view or table is defaulted to "Level". The Item Type for each database view or table column is defaulted to "Element".

If the source is based on a DTD or a production XML message, the default Item Type is "Element". DTDs do not support levels, therefore any levels must be explicitly defined by setting the Item Type to "Level".

Any node that represents a collection of data that repeats (for example, Purchase Order lines or shipment lines) represents a level. The Item Type for the node must be set to Level.

For the OAG standard, change the Item Type of the following to Level to support Level Mapping:

- Root
- CNTROLAREA
- DATAAREA

Make the same change for all other DTD data types identified as data levels during the map analysis process.

The Item Type for a new sibling or child element is defaulted to "Element". Change the Item Type setting when appropriate.

### Default

Enter a default value for the field as appropriate. This value is used in the outbound message or an inbound message if the incoming value is null.

If the default value is based on a condition, set the default using the Assign Variable Value action. See Assignments: Assign Variable Value, page 2-58 .

If Item Type is "Level", this column is disabled.

## Category

Enable the field for code conversion by entering a valid code category. Validation of code category is performed at runtime because the database used to define the map may not be the database where the transaction is executed. Refer to Seeded Code Categories, page B-1 for a list of seeded code categories.

Universal or standards-specific code conversion values are defined using the Define Code Conversion Values form. Trading Partner-specific code conversion values are defined using the Trading Partner Code Conversion form.

The execution engine will search for code conversion values in the following sequence:

- Trading Partner
- Standard-specific
- Universal list

Use the Get Predefined Variable Value action to determine the status of the code conversion process for the source column enabled for code conversion.

See Get Predefined Variable Value, page 2-76 for details regarding the code conversion return status and possible actions to take if the code conversion cross-reference value is not found.

Code conversion is applied before any Actions are applied. Consider the code conversion return status when you define Actions for the source column enabled for code conversion. Actions are applied to the code-converted value only when the code conversion process is successful.

If Item Type is Level, this column is disabled.

## Data Type

Each field is defined with a data type. The data types supported by the XML Gateway Execution Engine are VARCHAR2, DATE, NUMBER, CHAR, and CLOB.

The CLOB data type is used to support large objects up to 4 GB in size. The CLOB is displayed between CDATA tags to escape parsing.

If the source is based on database views or tables, the data types are defaulted to the data types defined for the view or table columns.

If the source is based on a DTD or a production XML message, the data type is defaulted to VARCHAR2.

**Important:** It is not necessary to change the DTD element's data type until XML schemas are supported by the XML standards bodies which will make data types more meaningful.

The Data Type for a new sibling or child element is defaulted to VARCHAR2. Change the data type if necessary.

If the Item Type is Level, the Data Type column is disabled.

## DB Column

The DB Column is available only when the source is based on database views or tables.

A check mark indicates the column is defined in the Oracle E-Business Suite data model. This setting informs the XML Gateway Execution Engine whether to validate the element against the Oracle E-Business Suite data model or not.

If Item Type is Level, the DB Column is disabled.

## Node Type

Node Type is available only when the source is a DTD. The default is the DTD setting.

The valid values are Element or Attribute. Elements contain the business data. Attributes contain qualifiers for the business data to indicate its intended meaning.

The Node Type for a new sibling or child element is defaulted to Element. Change the Node Type setting where appropriate.

If Item Type is Level, this column is disabled.

## Add Sibling (button)

The Add Sibling button allows you to add new fields required to complete the map. This feature is also used to create duplicate DTD nodes such as PARTNER.

When the source is based on a DTD or production XML message, and you are adding a sibling between two attributes, set the Node Type for the new field to "Attribute". The default Node Type of "Element" will cause a parser violation.

**Important:** You cannot add a sibling to the root node.

Refer to How to Extend DTDs, page 2-84 for details on how to extend a DTD. Included are the naming conventions that must be followed for the XML Parser to recognize the DTD extensions.

## Add Child (button)

The Add Child button adds child elements to an existing sibling or child. In addition, Add Child can be used to define an attribute for the root element if the source or target is a DTD.

This function may be used if the source is database views or tables or a DTD.

If Node Type is "Attribute", the Add Child button is disabled. You cannot define an attribute for an attribute.

**Note:** Refer to How to Extend DTDs, page 2-84 for details on how to extend a DTD. Included are the naming conventions that must be followed for the XML Parser to recognize the DTD extensions.

## Delete (button)

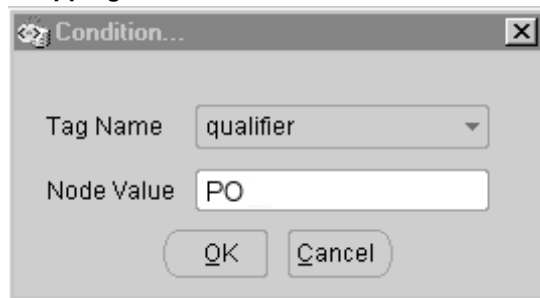
The Delete button allows you to delete any sibling or child element that has not been mapped.

If a sibling or child has child elements associated with it, a warning is displayed before the delete occurs.

**Important:** If you are deleting any DTD extensions, be sure to remove the DTD extensions from the corresponding extension file created for the application or user. The extra information will not cause a parser

violation, but it is a good practice to ensure the extension files match the message maps. Refer to *How to Extend DTDs*, page 2-84 for the details.

### Conditional Node Mapping Window



### Condition.../Delete Condition

If the source definition is based on a DTD with duplicate nodes or elements of the same name, use Conditional Node Mapping to ensure that the source-to-target mappings are performed correctly. This is required because the sequence of duplicate nodes and elements is not fixed.

Examples of duplicate nodes that occur frequently in a DTD are DATETIME, AMOUNT, OPERAMT, and QUANTITY. Using a purchase order as an example, the first DATETIME occurrence is for the Order Date and the second occurrence is for the Promise Date.

Conditional node mapping allows you to define the relationship of the source node to the target node based on the value of key elements. The key elements are identified by the Tag Name and Node Value combination. For the purchase order example above, the Tag Name is "qualifier" and the Node Value is "PO" for the first DATETIME occurrence representing the order date. The Tag Name for the second DATETIME occurrence is also "qualifier" with a Node Value of "PROMDELV" for promise date.

To define the key values, select the source DTD node and click the right mouse button to invoke the Conditional Node Mapping window. You will be prompted for the following:

#### Tag Name

Select Tag Name from the list of values.

Using the above example, this is the "qualifier" attribute associated with the DATETIME segment that uniquely identifies the node and its intended meaning.

#### Node Value

Enter the node value.

Using the above example, this is either "PO" or "PROMDELV".

Repeat this process for each of the duplicate nodes and elements. Message Designer supports one condition per node or element.

To delete the Conditional Node Mapping instruction for a node, select the node and click the right mouse button to invoke the Delete Condition function.

Conditional node mapping is applicable to source data definitions that are based on a DTD or a production XML message. It does not apply when the source data definitions are based on database views or tables.

## Transaction Map - Target Definition

The target definition selected or created using the Map Creation Wizard is displayed in the Target Definition Tab. The target definition can be extended to perform the following:

- Provide default values
- Create duplicate nodes and elements using the Add Sibling button
- Add fields using the Add Child button
- Delete unused and unmapped DTD elements to prevent parser violations

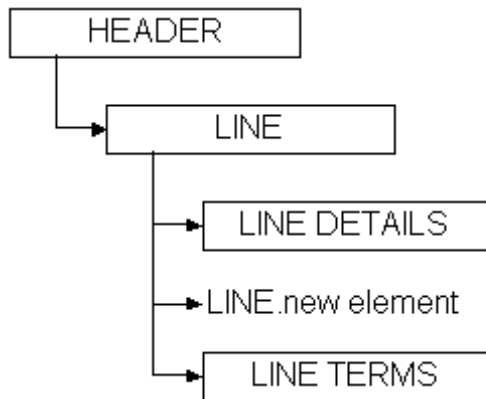
Once the target data definition is complete, it can be saved for reuse in other message maps. Use the File > Save (Data Definition) menu option or the toolbar equivalent. Use the File > Properties menu option to change the default directory or data definition property values if necessary.

If you do not wish to save the target data definition as an entity independent of the message map, then continue with the mapping process and save the transaction map at the end of the Element Mapping process.

### Target Definition Considerations

1. Refer to How to Map a Pass-Through Message, page 2-86, for guidelines related to developing a pass-through transaction.
2. Refer to How to Map to an API, page 2-86 for guidelines for mapping an inbound message to an Application API (as opposed to mapping to Application Open Interface tables).
3. For DTD elements defined using "|" as the occurrence indicator, make sure you select one element from the choice list and delete the unused elements. The parser will validate that only one element is used.
4. Delete optional DTD data types and elements that are not used and therefore not mapped. If not deleted, these elements will appear as empty tags in the resulting message.
5. If you decide not to delete DTD data types and elements that are not used and not mapped, default the data type or element attribute to "OTHER". The parser requires the attribute even though the data type or element is optional.
6. Refer to How to Implement Attachments in XML Messages, page 2-90 for details on how to include attachments with your XML documents.
7. Discontinuous nodes:

A discontinuous node is a non-level node that follows a level and is a sibling of that level. It can be represented graphically as follows:



In this example, the levels LINE DETAILS and LINE TERMS are children of the level LINE, and are siblings of each other. Inserted between LINE DETAILS and LINE TERMS is a new element of the LINE node. The new element is a continuation of the LINE node that creates a break between LINE DETAILS and LINE TERMS.

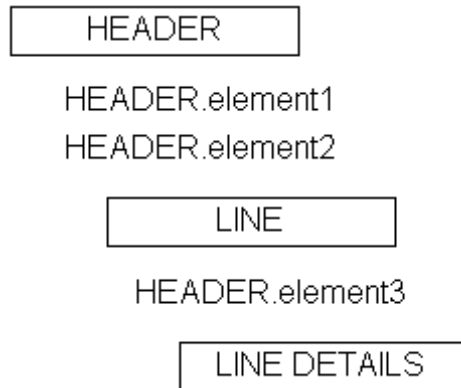
The ideal placement of the new element is as the last element of the LINE node before the LINE DETAILS node. However, some standards bodies do not have the flexibility to restructure an existing DTD, or may not wish to for backwards compatibility reasons. Regardless of the reason, the condition exists, and XML Gateway supports it.

For data definitions based on the Oracle data model, use the Add Child or Add Sibling button in the Message Designer, Transaction Map window to define a discontinuous node. Any new node introduced on the source and distributed across multiple target levels (expanded) or consolidated into a single target level (collapsed) will be grouped with the parent node and mapped according to the target definition. The rules against level cross-over still apply.

For data definitions based on a DTD, use the Transaction Map window, Item Type column to explicitly identify the data levels. This exercise may create a discontinuous node, which is not a problem unless you define the invalid scenario described below. The only user extensions supported for OAG DTDs are in the USERAREA. If you modify the DTD (using the Add Child or Add Sibling buttons) outside of the USERAREA, a parsing error will be triggered.

Message Designer will allow you to define a discontinuous node anywhere in the source or target definition. You will not know until runtime if the definition is valid or not. Therefore you should avoid introducing a discontinuous node for a node which also contains a child node.

The following graphic illustrates an invalid definition:



In the example above, HEADER.element3 is a continuation of the HEADER node. HEADER.element3 is also defined as a child of the LINE node. HEADER.element3 has a child node called LINE DETAILS. This is an invalid definition.

## Target Definition Tab

**Target**

- PROCESS\_INVOICE\_001
  - CNTRLAREA
    - DATAAREA
      - PROCESS\_INVOICE
        - INVOICEHDR
          - AMOUNT
          - DATETIME
          - INVOICEID
          - DESCRIPTN
          - DOCTYPE
          - PAYMETHOD
          - REF
          - REMITTANCE
          - USERID
          - VOUCHER
          - USERAREA

**Target Elements**

	Field	ItemType	Default	Data Type
1	PROCESS_INVOICE_...	Level		
2	CNTRLAREA	Element		VARCHAR
3	BSR	Element		VARCHAR
4	VERB	Element		VARCHAR
5	value	Element	PROCESS	VARCHAR
6	NOUN	Element		VARCHAR
7	value	Element	INVOICE	VARCHAR
8	REVISION	Element		VARCHAR
9	value	Element	001	VARCHAR
10	SENDER	Element		VARCHAR
11	LOGICALID	Element		VARCHAR
12	COMPONENT	Element		VARCHAR
13	TASK	Element		VARCHAR
14	REFERENCEID	Element		VARCHAR
15	CONFIRMATION	Element		VARCHAR

## Field

Field identifies the name of the element, document, or root. The names are based on the Application Open Interface table column names or DTD element names.

The field name can be changed if necessary, however, consider what this change implies. Because the field is based on the database column name or a DTD element name, corresponding changes may be necessary in Oracle E-Business Suite or the DTD. The only changes allowed to a DTD are to the USERAREA. Refer to How to Extend DTDs, page 2-84 for details.

The first row is reserved for the Data Definition name (if target is based on Application Open Interface) or DTD root element name (if target is a DTD).

Additional field names are displayed when sibling or child elements are added using the Add Sibling or Add Child buttons.

## Item Type

Item type identifies the field as either a Level or an Element. Level represents the parent in a parent-child relationship. Element represents the child in a parent-child relationship.

The Item Type of the first row is defaulted to Level. The default value cannot be changed.

If the target is based on Application Open Interface tables, the Item Type for the tables is defaulted to "Level". The Item Type for the columns is defaulted to "Element".

If the target is based on a DTD or a production XML message, the default Item Type is "Element". DTDs do not support levels and therefore the levels must be explicitly defined by setting the Item Type to "Level".

Any node that represents a collection of data that repeats (for example, PO lines or shipment lines) represents a level. The Item Type for the node must be set to "Level".

For the OAG standard, change the Item Type of the following to "Level" to support Level Mapping:

- Root
- CNTRLAREA
- DATAAREA

Make the same change for all other DTD data types identified as data levels during the map analysis process.

The Item Type for a new sibling or child element is defaulted to "Element". Change the Item Type setting where appropriate.

## Default

Enter a default value for the field as appropriate. This value is used in the outbound message. The default value will be used in an inbound message if the incoming value is null.

If the target is a DTD, use the Default column to set the DTD attribute values. The values will be displayed with the corresponding attribute tags when the message is created.

If the default value is based on a condition, set the default using the Assign Variable Value action. See Assignments: Assign Variable Value, page 2-58.

If Item Type is Level, this column is disabled.

## Data Type

Each field is defined with a data type. The data types supported by the XML Gateway Execution Engine are VARCHAR2, DATE, NUMBER, CHAR, and CLOB.

The CLOB data type is used to support large objects up to 4 GB in size. The CLOB is displayed between CDATA tags to escape parsing.

If the target is based on Application Open Interface tables, the data type is defaulted to the data type defined for the database column.



If the target is based on a DTD or a production XML message, the data type is defaulted to VARCHAR2.

**Important:** It is not necessary to change the DTD element's data type until XML schemas are supported by the XML standards bodies which will make data types more meaningful.

The Data Type for a new sibling or child element is defaulted to VARCHAR2. Change the data type if necessary.

If Item Type is Level, the Data Type column is disabled.

## DB Column

DB Column is available only when the target is based on Application Open Interface tables.

A check mark indicates the column is defined in the Oracle E-Business Suite data model. This setting informs the XML Gateway Execution Engine whether to validate the element against the Oracle E-Business Suite data model or not.

If the Item Type is Level, the DB Column is disabled.

## Node Type

Node Type is available only when the target is a DTD. The default is the DTD setting.

The valid values are Element or Attribute. Elements contain the business data. Attributes contain qualifiers for the business data to indicate the intended meaning of the data.

The Node Type for a new sibling or child element is defaulted to "Element". Change the Node Type setting if appropriate.

If Item Type is Level, the Node Type column is disabled.

## Add Sibling (button)

The Add Sibling button allows you to add new fields required to complete the map. This feature is also used to create duplicate DTD nodes such as PARTNER.

The same can be done if the target is Application Open Interface tables.

When the target is based on a DTD or production XML message, and you are adding a sibling between two attributes, set the Node Type for the new field to "Attribute". The default Node Type of "Element" will cause a parser violation.

**Important:** You cannot add a sibling to the root node.

Refer to How to Extend DTDs, page 2-84 for details on how to extend a DTD. Included are the naming conventions that must be followed for the XML Parser to recognize the DTD extensions.

## Add Child (button)

The Add Child button adds child elements to an existing sibling or child. In addition, Add Child can be used to define an attribute for the root element if the source or target is a DTD.

This function can be used if the target is Application Open Interface tables or a DTD.

If Node Type is "Attribute", the Add Child button is disabled. You cannot define an attribute for an attribute.

**Note:** Refer to How to Extend DTDs, page 2-84 for details on how to extend a DTD. Included are the naming conventions that must be followed for the XML Parser to recognize the DTD extensions.

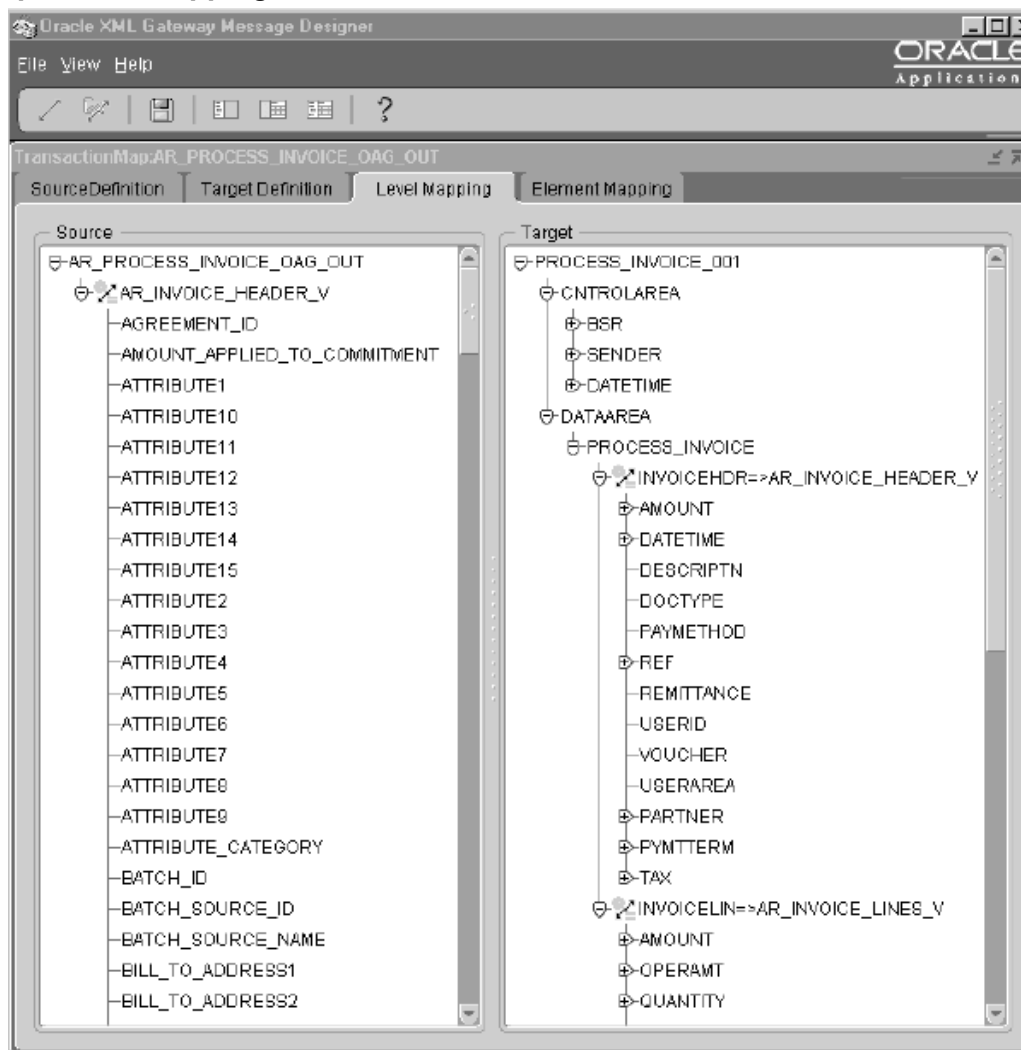
### Delete (button)

The Delete button deletes any sibling or child element that has not been mapped.

If a sibling or child has child elements associated with it, a warning is displayed before the delete occurs.

**Important:** If you are deleting any DTD extensions, be sure to remove the DTD extensions from the corresponding extension file created for the application or user. The extra information will not cause a parser violation, but it is a good practice to ensure the extension files match the message maps. Refer to How to Extend DTDs, page 2-84 for the details.

## Transaction Map - Level Mapping Tab



In the Level Mapping tab, the source definition is presented in the left window and the target definition in the right window. Use the Level Mapping tab to relate the source hierarchy to the target hierarchy. All entities defined as a level are displayed in bold.

Select the source level and drag it to the desired target level. The source level name and the mapped icon will appear to the right of the target level name indicating the level is mapped.

If this does not occur as described, you have not set the DTD entity Item Type to "Level". Return to the Source or Target Definition tab and set the Item Type accordingly, then resume with Level Mapping.

To unmap a mapped level, simply select the mapped level on the target window and drag it back to the source level.

### Level Mapping Guidelines for OAG DTDs

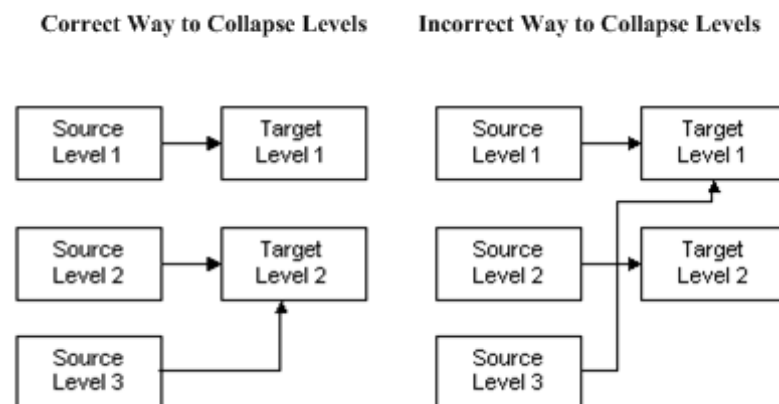
1. For the outbound messages using the OAG standard, map the ECX\_OAG\_CONTROLAREA\_TP\_V view (formerly ECX\_OAG\_CONTROLAREA\_V) to the DTD CNTROLAREA data type. This step is not required for inbound messages because the content of the DTD CNTROLAREA is not stored in the Oracle E-Business Suite.

**Note:** The ECX\_OAG\_CONTROLAREA\_TP\_V view is an upgraded version of the ECX\_OAG\_CONTROLAREA\_V view. Oracle XML Gateway supports both versions of the database view. For a detailed description of the differences, see the Note, page 2-25, presented earlier in this chapter.

2. If you anticipate multiple documents in a message, map the database header view to the OAG document header data type. If you anticipate a single document in a message, map the database header view to the OAG DATAAREA data type.

### Level Mapping Guidelines to Collapse Levels

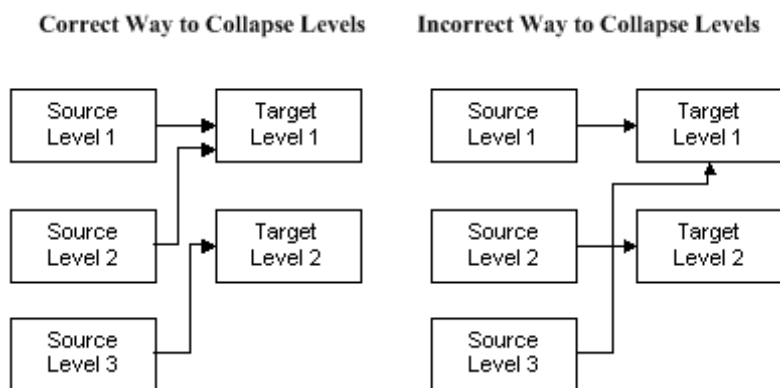
Multiple source levels can be mapped to the same target level. This is commonly referred to as collapsing levels. For example, if your source is 3 levels and your target is 2 levels you can collapse the levels as shown in the following figure:



In the correct example above, there are three source levels and two target levels. Source Level 1 is mapped to Target Level 1. Source Levels 2 and 3 are mapped to Target Level 2. The result of collapsing levels is that the data in Source Levels 2 and 3 are consolidated and mapped to Target Level 2. If there are two rows in Source Level 2 and three rows in Source Level 3, a total of six rows will be created in Target Level 2.

In the example showing the incorrect collapsing of levels, Source Level 3 is mapped over Target Level 2 to Target Level 1.

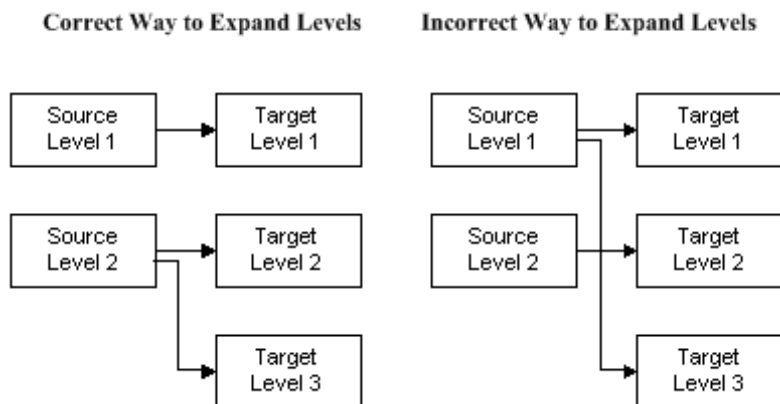
Another option shown below is to relate Source Levels 1 and 2 to Target Level 1 and relate Source Level 3 to Target Level 2. (Do **not** map Source Levels 1 and 3 to Target Level 1, crossing over Target Level 2.)



Whichever option you choose, consider what it means to promote lower level detail data to a higher level. The lower level data may need to be aggregated to be meaningful at the higher level.

## Level Mapping Guidelines to Expand Levels

One source level can be mapped to multiple target levels. This is commonly referred to as expanding levels. For example, if your source is two levels and your target is three levels you can expand the levels as shown in the following figure:

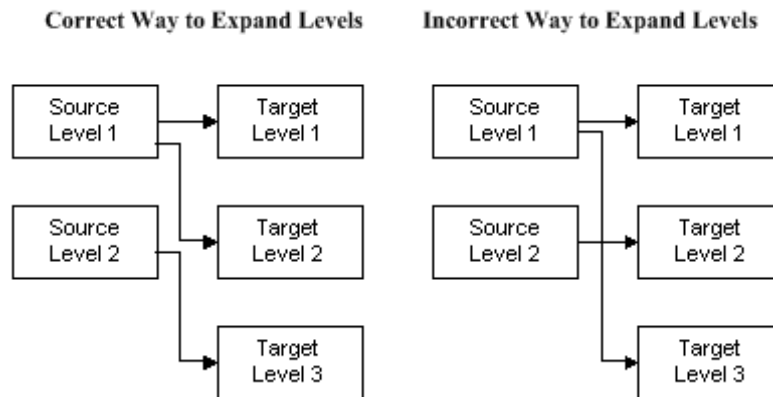


In the correct example above, Source Level 1 is mapped to Target Level 1 and Source Level 2 is mapped to Target Levels 2 and 3. The result of expanding levels is that the data

in Source Level 2 is distributed and mapped to Target Levels 2 and 3. If there are two rows in Source Level 2, two rows will be created in Target Level 2 and Target Level 3.

In the example showing the incorrect expansion of levels, Source Level 1 is mapped to Target Level 1 and Target Level 3, crossing over Target Level 2.

Another option, shown below, is to distribute Source Level 1 to Target Levels 1 and 2 and map Source Level 2 to Target Level 3. (Do **not** map Source Level 1 to Target Levels 1 and 3, crossing over Target Level 2.)

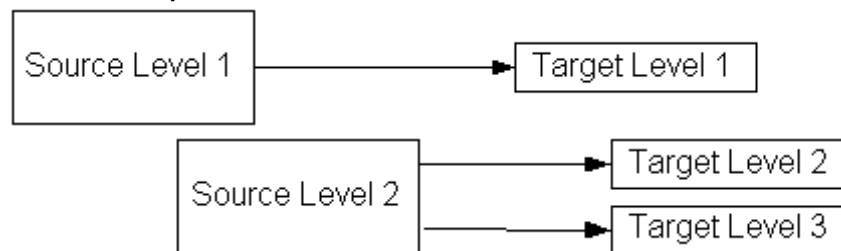


Whichever option you choose, consider what it means to demote data from a higher level to a lower level of detail. The higher level data may need to be deaggregated to be meaningful at the lower level.

### Level Expansion for Discontinuous Nodes

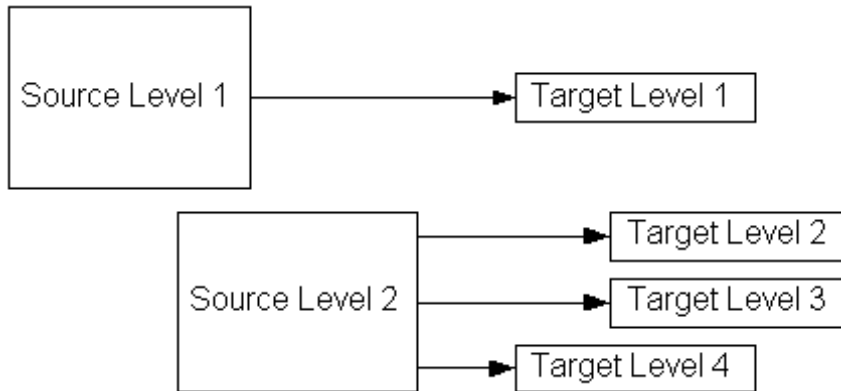
Level expansion is supported if the target expanded levels are all siblings of each other or if they are all children of the previous node.

#### **Valid Level Expansion**



In the example above, Target Level 2 and Target Level 3 are siblings to each other and children of Target Level 1.

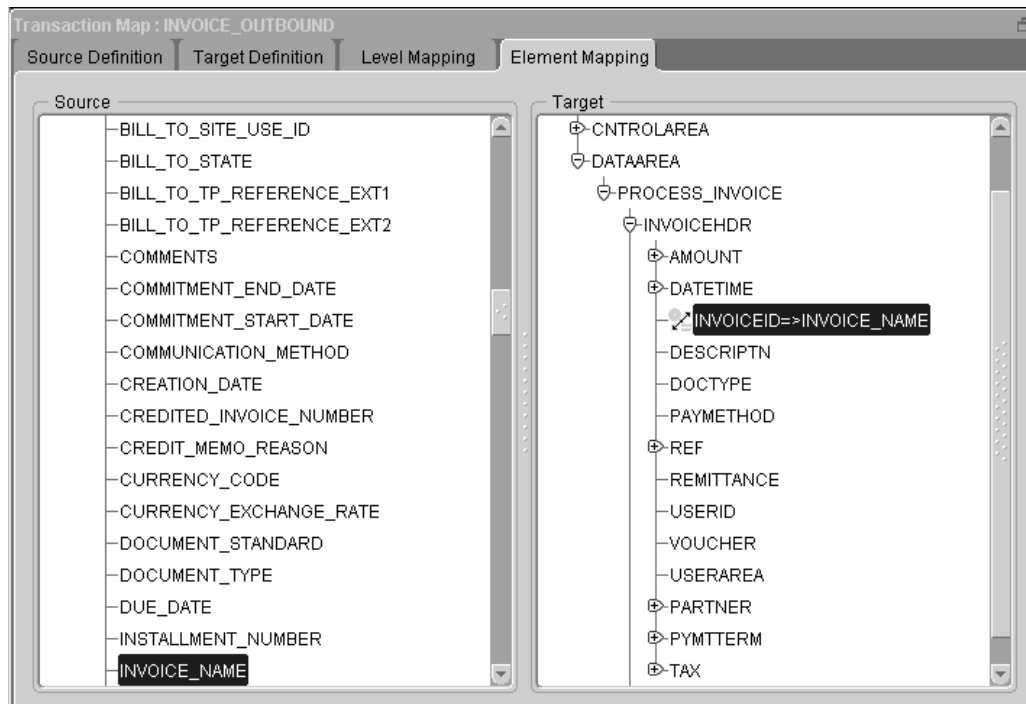
### Invalid Level Expansion



In the example of invalid level expansion above, Target Level 2 and Target Level 3 are siblings to each other and children of Target Level 1. Target Level 4 is a sibling of Target Level 1, with no relationship to Target Levels 2 and 3.

See Discontinuous Nodes, page 2-33 for more information on discontinuous nodes.

## Transaction Map - Element Mapping



If the Element Mapping tab is not available, it is an indication that you have not completed at least one level mapping in the Level Mapping process.

Once the Level Mapping process is complete, the source definition is presented on the left window and the target definition is presented on the right window. The source and target hierarchies defined in the Level Mapping tab are displayed in bold font. Use the Element Mapping tab to create the message map.

Select the source element and drag it to the target element. Pay special attention to any predefined node conditions (such as Conditional Node Mapping) to ensure that duplicate nodes are mapped to the correct target entities. The source element name and mapped icon will appear next to the target element name indicating the element is mapped.

To unmap a mapped element, select the mapped element on the target window and drag it back to the source element.

## Element Mapping Guidelines

1. Do not map a lower level element to a higher level element. The lower level element values are not available until the header level elements are completely processed.
2. A higher level element can be mapped to a lower level element. However, you may need to manipulate the higher level element value to make it meaningful in the context of a detail level element.

For example, if the the header level element is invoice total and the line level element is invoice line total, the header invoice total must be distributed across the invoice lines based on the line quantity for it to be meaningful in the context of invoice lines.

3. One source element can be mapped to multiple target elements.
4. Review the source elements containing Conditional Node Mapping instructions. Map the source element to the target element associated with the condition.

For the outbound messages (where the source is the database and the target is a DTD) using the OAG standard, map the ECX\_OAG\_CONTROLAREA\_TP\_V view (formerly ECX\_OAG\_CONTROLAREA\_V) columns to the DTD CNTROLAREA data type elements. This step is not required for inbound messages (where the source is a DTD and the target is the database) because the content of the DTD CNTROLAREA is not stored in the Oracle E-Business Suite.

**Note:** The ECX\_OAG\_CONTROLAREA\_TP\_V view is an upgraded version of the ECX\_OAG\_CONTROLAREA\_V view. Oracle XML Gateway supports both versions of the database view. For a detailed description of the differences, see the Note, page 2-25, presented earlier in this chapter.

The view column names are similar to the DTD CNTROLAREA data type element names, so this is one-to-one mapping. Add the following required Actions to complete the map:

- Use the Convert to OAG DATETIME action to convert the Oracle date to the CNTROLAREA DATETIME element.
- Use the Append Where Clause action to bind the transaction type and transaction subtype to the ECX\_OAG\_CONTROLAREA\_V view. Or if you are using the ECX\_OAG\_CONTROLAREA\_TP\_V view, use the Append Where Clause to bind the transaction type, transaction subtype, party ID, party site ID, and party type to the view.

**Note:** The ECX\_OAG\_CONTROLAREA\_TP\_V view is an upgraded version of the ECX\_OAG\_CONTROLAREA\_V view. Oracle XML Gateway supports both versions of the database view. For a detailed description of the differences, see the Note, page 2-25, presented earlier in this chapter.

Use the Define Transactions form to define the transaction and transaction subtype that represent the Oracle name for the message. Associated with the internal name for the message are the external type and subtype that represent the message name in the XML standard of choice.

For OAG, the external subtype corresponds to the BSR VERB and the external type corresponds to the BSR NOUN. The names entered on the Define Transactions form are stored in the database and accessed by the ECX\_OAG\_CONTROLAREA\_TP\_V view. The values are displayed in the Message Designer as default values for the BSR VERB and BSR NOUN elements.

Refer to the Define Transactions Form, page 3-7 for the details and observe the recommended naming conventions.

## Element Mapping Icons

Message Designer uses an icon group to help you determine the status of a map at a glance. The components of the icon group are as follows:



Mapped Element icon.



Action Defined icon.



Code Conversion Enabled icon.

The components will be displayed as on or off (grayed out) within the group icon to indicate if the element is mapped, has an action defined, and/or is enabled for code conversion.

## Element Mapping and Actions

As part of the element mapping process, Actions for data transformation and process control can be defined.

The following three sections are available for your reference. First time users should read all three sections. Experienced users should use the Map Action Editor section and reference the action type of interest.

- Transaction Map - Actions, page 2-51
  - Summary of XML Gateway Supported Actions, page C-1
  - Source or Target Action, page 2-51
  - Action Levels, page 2-52
  - XML Gateway Execution Engine processing sequence, page 2-52
- Map Action Editor - Overview, page 2-53
  - How to Invoke the Map Action Editor, page 2-53



- Map Action Editor Components, page 2-53
- Map Action Editor - Available Actions, page 2-54

At the completion of the element mapping process, use the File > Save (Transaction Map) menu option or the Save icon to save the map onto the file system. The name of the map (.xgm) file should be the same as the map name for easy reference. Follow the map naming conventions as follows:

- Product mnemonic or user ID
- Transaction subtype as entered in the Define Transactions form
- XML standard and version used (such as OAG, Rosettanet, iFX)
- Outbound or inbound message

Given a map name of:

AR\_INVO\_OAG70\_OUT

the map file name is:

AR\_INVO\_OAG70\_OUT.xgm

Use the File > Properties menu option to change the default directory or map name if necessary.

The transaction map and the associated DTDs are now ready to load into the XML Gateway repository. The message maps will be used by the XML Gateway Execution Engine to create or to consume XML messages.

Refer to How to Load/Delete Message Maps and DTDs, page 2-87 for details on loading a message map created by the Message Designer and its associated DTDs into the XML Gateway repository.

## Transaction Map - Actions

As part of the element mapping process, Actions for data transformation or process control can be defined.

Actions are similar to prebuilt functions in that they may be called to perform a specific activity.

Oracle XML Gateway supports actions for data transformation involving math functions, string manipulation, and data conversion between Oracle's data format and OAG's data format.

Oracle XML Gateway supports a set of predefined actions for process control. This includes user-defined procedure and function calls to extend the integration with the Oracle E-Business Suite. Other common process control actions allow you to inquire on the status of a transaction and to manage the process flow based on the status. For serious errors, the transaction can be aborted with error messages returned to the sender via an Oracle Workflow process.

The actions supported by Oracle XML Gateway are listed in Appendix C, page C-1.

## Source or Target Action

Most Actions are defined on the target side of the Element Map with the exception of the Append Where Clause action type, which is defined on the source side of the Element Map if the source is based on database views or tables.

See Map Action Editor - Append Where Clause, page 2-60 for a detailed description of this action and how to define it.

## Action Levels

An Action may be defined for any of the following entities:

- Element

An element is the smallest unit of a message. An action defined at the element level is applied to that element only.

- Document

A document is a collection of elements representing a business document. An action defined at the document level is applied to the document.

- Root

A root represents a collection of documents. An action defined at the root level is applied to all documents contained by the root.

Some actions are designed to be applied to the element only, while others are intended for the document only.

## XML Gateway Execution Engine Processing Sequence

The processing sequence of a three-level document by the XML Gateway Execution Engine is described in the following table.

Refer to Pre Process, In Process, and Post Process Tabs, page 2-54, for a description of process stages.

Level	Stage
Root	Preprocess
Root	In-Process
Header	Preprocess
Header	In-Process
Header	Postprocess
Line	Preprocess
Line	In-Process
Line	Postprocess
Line Detail	Preprocess
Line Detail	In-Process
Line Detail	Postprocess
Root	Postprocess

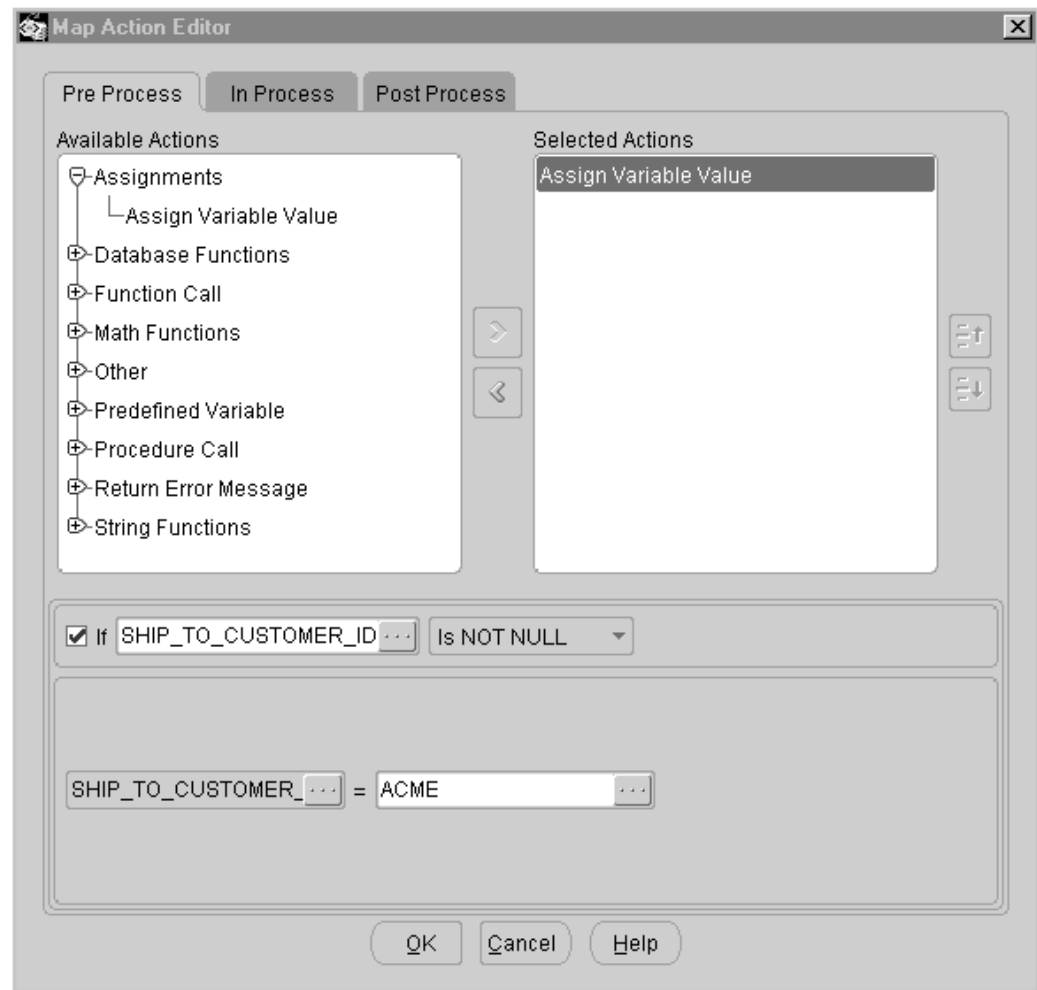
Given the processing sequence described above, header data is processed before line data and line data is processed before detail data. Any upper level data element that has a dependency on lower level data elements (for example, sum of invoice lines) must be

processed using an API call at the upper level. Once the upper level data is processed, it cannot be updated, although it can always be accessed.

## Map Action Editor

### How to Invoke the Map Action Editor

In the Element Mapping tab, select the entity (element, document, root) and click the right mouse button to invoke the Map Action Editor pop-up window. The prompts presented by the Map Action Editor will vary based on the Action type selected.



### Overview

The Map Action Editor has the following common components:

- Pre Process, In Process, Post Process Tabs
- Available Actions
- Selected Actions
- Optional Conditional Expression

- Result Variable
- Action Operands - Element Window

### **Pre Process, In Process and Post Process Tabs**

An Action can be applied at any of the following stages of message creation or consumption:

- Pre Process

A preprocess action is executed before the message is created or consumed.

The Create Global Variable action is an example of a preprocess action. The variable must be defined before you can use it.

- In Process

An in-process action is executed during message creation or consumption.

The Math and String Functions are examples of in-process actions used to perform a computation or to manipulate a value.

- Post Process

A postprocess action is executed after the message is created or consumed.

The Insert into Database Table action is an example of a postprocess action. The row cannot be inserted into the database until all the data for the row has been processed.

Select the appropriate Map Action Editor Window Tab for the map entity (element, document, root) selected.

### **Available Actions**

The available action categories for the selected process stage and selected map entity are displayed. Expand each category to view all the available action types.

The following table summarizes all action types sorted by Action Level and Action Stage. A "Y" indicates the action type is available for the Action Level and Action Stage combination. An "N" indicates the action type is not available for the Action Level and Action Stage combination.

Action Category & Description	Element In Process	Document Pre Process	Document In Process	Document Post Process	Root Pre Process	Root In & Post Process
<b>Assignment:</b> Assign variable value	Y	Y	Y	Y	N	Y
<b>Assignment:</b> Create global variable	N	N	N	N	Y	N
<b>Database Function:</b> Assign next sequence value	Y	Y	Y	Y	N	Y
<b>Database Function:</b> Append where clause	N	DB Source	N	N	N	N
<b>Database Function:</b> Insert into database table	N	N	N	Y	N	N
<b>Derivations</b>	Y	N	N	N	N	N
Function Call	Y	Y	Y	Y	N	Y
Math Functions	Y	Y	Y	Y	N	Y
OAG Standard Conversions	Y	N	N	N	N	N
<b>Other:</b> Exit Program	Y	Y	Y	Y	N	Y
<b>Predefined Variable</b>	Y	Y	Y	Y	N	Y
Procedure Call	Y	Y	Y	Y	N	Y
<b>Return Error Message</b>	Y	Y	Y	Y	N	Y
String Functions	Y	Y	Y	Y	N	Y
<b>XSLT Transformation</b>	N	N	N	N	N	Y(Post)

### Selected Actions

To move an action type from Available Actions to Selected Actions, select the desired action type and click the shuttle button.

To deselect a selected action type, select the desired action type and click the shuttle button to move the selected action type from Selected Actions back to Available Actions.

## Up and Down Arrows for the Selected Actions Window

A given entity (element, document, root) may have one or more Actions defined for it. The XML Gateway Execution Engine processes the Actions in the sequence defined, from the beginning of the list proceeding downward to the last action type on the list. Use the up and down arrows within the Selected Actions Window to change the sequence of the Actions defined for a given entity. Use the left shuttle button to deselect a Selected Action if necessary.

## Optional Condition Expression: If <Operand 1><Operator><Operand 2>

Each Action may be defined as a conditional action (except the Append Where Clause and Create Global Variable). The condition is expressed as two operands. An operand may be a variable (source, target, or global variable) whose value is determined at runtime, or a literal value.

For each condition operand, click on the (...) icon to the right of the operand field to invoke the Element window. The Element window allows you to select a variable or provide a literal value.

Refer to Action Operands - Element Window, page 2-57 for details regarding the Element window.

The operators supported by Oracle XML Gateway are listed in the following table:

Operator	Description
=	Equal
!=	Not Equal
>	Greater Than
<	Less Than
>=	Greater Than or Equal To
<=	Less Than or Equal To
Is NULL	Null
Is NOT NULL	Not Null

Enter the two operands and select the operator if a condition expression is required.

Compound conditions can be defined by comparing two operands and storing the result in a variable (source, target, or global). You then compare the value in the variable with the next condition operand.

## Result Variable

Some Actions have result variables defined and some do not. Those that do not have a result variable defined will use one of the action operands as the result variable.

The result variable is the left-most field of the action operand. The only exception occurs in the Convert To actions where the result variable is the right-most field. Per user interface standards, the result variable field is greyed, but is enabled for data entry.

For element-level Actions, the selected source or target variable is displayed as the default result variable. You can change the variable name as necessary.

For document-level or root-level actions, the result variable is blank. Click on the (...) icon to invoke the Element window. Select the source, target, or global variable as the result variable. The value of the result variable is determined at runtime.

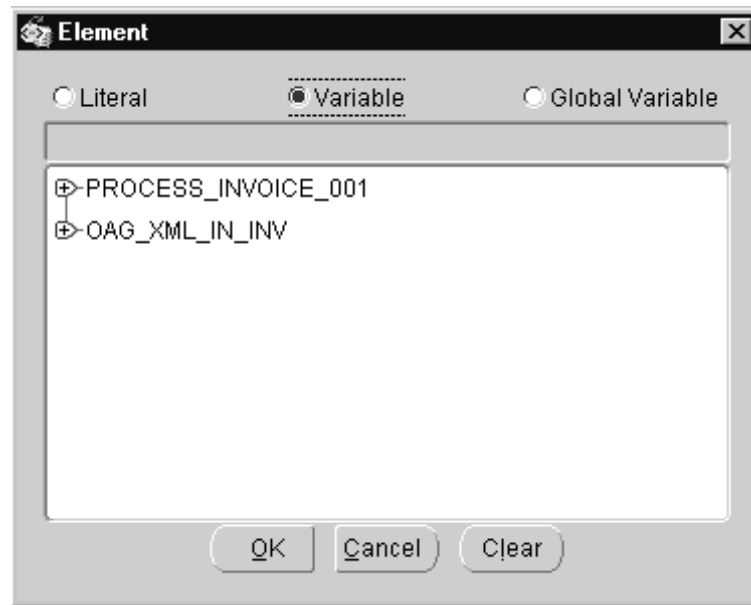
The result variable cannot be a literal value.

Refer to Action Operands - Element Window, page 2-57 for details regarding the Element window.

### Action Operands - Element Window

An Action may or may not have operands associated with it.

For Actions that do have operands associated with them, the number of operands varies by action type. Click on the (...) icon to the right of the operand field to invoke the Element window.



The Element window allows you to select a variable or provide a literal value. The Element window is presented in four parts as follows:

- Radio button for Literal, Variable, or Global Variable
- Operands displayed in gray indicate literal values are not allowed
- Field for literal value (available only when Literal radio button is selected)
- Window display for source and target variables

The default radio button setting is Variable. You can change the default by selecting a Literal (if available for the action type selected), or Global Variable (if defined).

The Literal radio button is enabled only if a literal value is applicable for the action type selected and the Element Window was not invoked from a Result Variable. Enter the literal value and click OK.

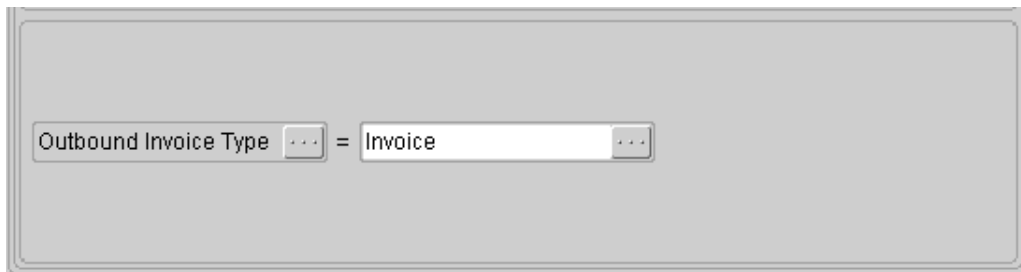
When the Variable radio button is enabled, you are presented with the source and target variables. Select the source or target variable name and click OK. The value for the selected variable is determined at runtime.

The Global Variable radio button is enabled only if global variables are defined for the message map. Select the Global Variable from the list of values and click OK. The value for the selected global variable is determined at runtime.

## Map Action Editor - Assignments: Assign Variable Value

The Assign Variable Value action assigns a value to the result variable identified. The value may be based on another variable (source, target, or global variable) or a literal value.

If you are assigning a literal value as a default value to be applied universally (without a condition expression), you have the option to set the default value in the Transaction Map form using the Source or Target Definition tab.



See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Result Variable

Select the source, target, or global variable to store the assigned value.

### Action Operands

Select the variable (source, target, or global variable) or enter the literal value to assign to the result variable.

## Map Action Editor - Assignments: Create Global Variable

The Create Global Variable action allows you to define a variable available to both the source and target. A default value may be assigned to the variable or the variable may get its value from the Assign Variable Value action.

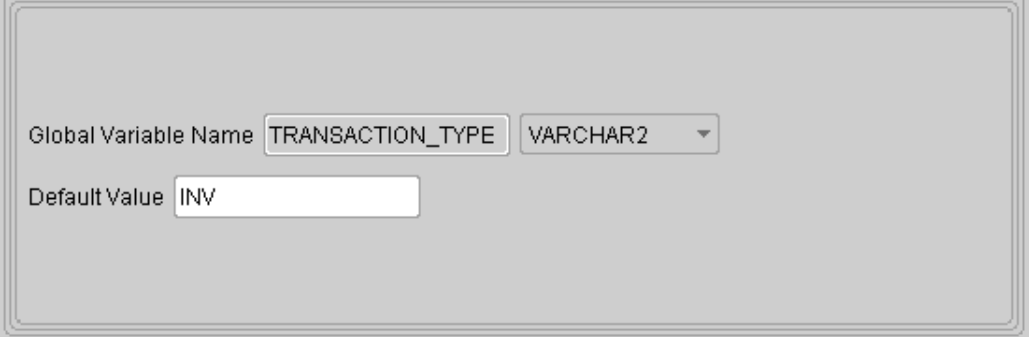
Once a global variable is defined, it is included in the Element window Global Variable Name list of values.

**Note:** The following are reserved names for global variables:

- TRANSACTION\_TYPE
- TRANSACTION\_SUBTYPE
- DOCUMENT\_ID
- TP\_SITE\_ID (also known as PARTY\_SITE\_ID)
- TP\_ID (also known as PARTY\_ID)
- TP\_TYPE (also known as PARTY\_TYPE)



- PARAMETER1, PARAMETER2, PARAMETER3, PARAMETER4, and PARAMETER5



The screenshot shows a window titled 'Global Variable Name' with a text input field containing 'TRANSACTION\_TYPE' and a dropdown menu set to 'VARCHAR2'. Below this, there is a 'Default Value' label and a text input field containing 'INV'.

See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

The Optional Conditional Expression is not available for the Create Global Variable action because the action is a preprocess action and therefore data is not available yet.

### Global Variable Name

Enter a global variable name and select the data type for the variable. The valid data types supported by the XML Gateway Execution Engine are VARCHAR2, NUMBER, DATE, and CHAR.

**Important:** Define a unique and meaningful name that does not contain a reserved word and is not a predefined variable. Once the variable name is defined and stored for the map, it cannot be changed.

However, if necessary, you can delete the Create Global Variable action containing the incorrect variable name and then add a new action containing the correct name.

**Note:** The CLOB data type defined to support large objects (up to 4GB) is supported by the XML Gateway execution engine, but not by the Create Global Variable action.

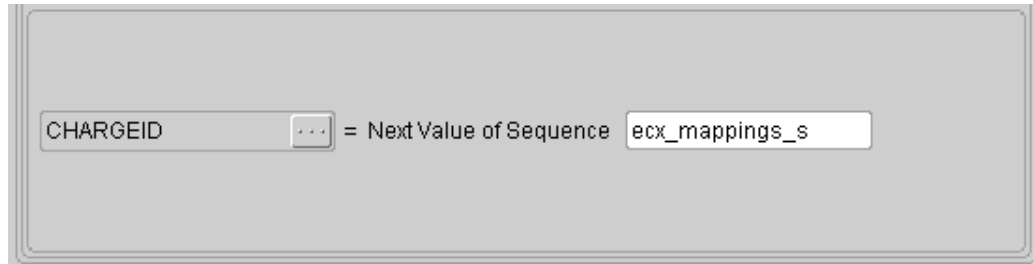
### Default Value

Enter a default value if applicable.

## Map Action Editor - Database Functions: Assign Next Sequence Value

Oracle E-Business Suite defines database sequences to maintain counters for document numbers such as PO or invoice number. When inserting documents into the Oracle Application Open Interface tables, the next available document number is required.

The Assign Next Sequence Value action retrieves the next available sequence number from the database sequence identified and assigns it to the result variable.



See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Result Variable

Select the source, target, or global variable to store the next value of the sequence identified.

### Next Value of Sequence

Identify the database sequence name. The function will assign the next value from the sequence to the result variable.

## Map Action Editor - Database Functions: Append Where Clause

The Append Where Clause action is used for outbound messages only. It is used to pass the document selection criteria to the database views used in the transaction map.

The Append Where Clause action is defined in the source at the document level as a preprocess activity. The bind variables and bind values are set up in advance of the document being processed. The actual bind to the database views occurs as an in-process activity when the data becomes available. The default where clause of "where 1=1" is appended at runtime to dynamically construct the where clause based on the selection criteria provided.

Each Append Where Clause action accepts one set of a bind variable and a bind value. To pass multiple selection criteria, you must define multiple Append Where Clause actions.

The document selection criteria are identified in the event and event subscription.

If you are using the ECX\_OAG\_CONTROLAREA\_V view in your map, you will need to bind to the TRANSACTION\_TYPE and TRANSACTION\_SUBTYPE columns for this view to execute properly.

If you are using the ECX\_OAG\_CONTROLAREA\_TP\_V view in your map, you will need to bind to the TRANSACTION\_TYPE, TRANSACTION\_SUBTYPE, PARTY\_ID, PARTY\_SITE\_ID, and PARTY\_TYPE columns for this view to execute properly.

**Note:** You must add the ECX\_EVENT\_MESSAGE item attribute to your Workflow item type to have access to the event details stored in the REFERENCE ID column of the ECX\_OAG\_CONTROLAREA\_TP\_V view.

The following table shows an example:

Where Clause	Bind Variable	Bind Value
and ECX_OAG_CONTROLAREA_TP_V. TRANSACTION_TYPE=:TTYPE	TYPE	TRANSACTION_TYPE
and ECX_OAG_CONTROLAREA_TP_V. TRANSACTION_SUBTYPE=:SUBTYPE	SUBTYPE	TRANSACTION_SUBTYPE
and ECX_OAG_CONTROLAREA_TP_V.PARTY_ ID=:PARTY_ID	PARTY_ID	TP_ID
and ECX_OAG_CONTROLAREA_TP_V.PARTY_S ITE_ID=:PARTY_SITE_ID	PARTY_SITE_ID	TP_SITE_ID
and ECX_OAG_CONTROLAREA_TP_PARTY_TY PE=:PARTY_TYPE	PARTY_TYPE	TP_TYPE
and <APPS_HEADER_V>.<document id>=:DOCID	DOCID	document_id

**Note:** For this method, use the Create Global Variable action to define variables to store the bind value. The global variables must be defined with the exact spelling of the parameters. In this example, the global variables are TRANSACTION\_TYPE, TRANSACTION\_SUBTYPE, TP\_ID, TP\_SITE\_ID, TP\_TYPE, and DOCUMENT\_ID.

The following are reserved names for global variables:

- TRANSACTION\_TYPE
- TRANSACTION\_SUBTYPE
- DOCUMENT\_ID
- TP\_SITE\_ID (also known as PARTY\_SITE\_ID)
- TP\_ID (also known as PARTY\_ID)
- TP\_TYPE (also known as PARTY\_TYPE)
- PARAMETER1, PARAMETER2, PARAMETER3, PARAMETER4, and PARAMETER5

Other source variables not related to the event triggering process can be used as bind values to ensure that the correct document is selected from the Oracle E-Business Suite database.

If you know the key values, you can define them in your where clause. With this method, you do not have to use the Bind Variable and Bind Value. The following table shows an example:

Where Clause	Bind Variable	Bind Value
and ECX_OAG_CONTROLAREA_TP_V.TRANSACTION_TYPE='POO'	N/A	N/A
and ECX_OAG_CONTROLAREA_TP_V.TRANSACTION_SUBTYPE='POOB'	N/A	N/A
and ECX_PO_HEADER_V.PO_NUMBER='A754739'	N/A	N/A

**Note:** Literal strings must be bound by single quotes.

Where Clause

Bind Variable

Bind Value

See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

The Optional Conditional Expression is not available for the Append Where Clause action because it is a preprocess action and therefore the data is not available yet.

### Where Clause

Identify the where clause to bind the database views used for the outbound message. This where clause will be appended to the default where clause of "where 1=1".

If you have multiple selection criteria based on the same database view, define an Append Where Clause action for each of the criteria. Define all fields for the first Append Where Clause action (Where Clause, Bind Variable, and Bind Value). For the subsequent Append Where Clause actions, define only the Bind Variable and Bind Value (leaving the Where Clause blank).

### Bind Variable

Identify the bind variable to be used by the where clause.

### Bind Value

Identify the bind value to be assigned to the bind variable used by the where clause.

The bind value is a global variable whose value is determined at run time, or a literal value.

## Map Action Editor - Database Functions: Insert into Database Table

The Insert into Database Table action is a postprocess action used for inbound messages to insert the data into Application Open Interface tables identified in the message map. This action is available only if the target is database.

This action must be executed once for each target level identified in the Transaction Map - Level Mapping tab. If the target has three levels (header, line, and line detail), then it must be executed three times.

You can identify a condition for the Insert into Database Table action. The action does not require any operands.

The inserts are performed by the XML Gateway Execution Engine but are not committed to the database until the entire document is processed. This eliminates the possibility of inserting a partial document into the database.

**Note:** An alternate way to insert data is to use the Procedure Call action to execute an Application API. See How to Map to an API, page 2-86 for details.

See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

## Map Action Editor - Derivations: Derive Address ID from Location Code

The Derive Address ID from Location Code action is used for inbound messages only. It uses the source location code and associated address type to derive the address and organization ID meaningful to the target Oracle E-Business Suite module.

The IDs are meaningful in the context of Oracle E-Business Suite only. The sender is not expected to know a valid address or organization ID.

If the parent table ID for the customer or supplier site is also required by the Application Open Interface, use the Derive Parent ID from Location Code, page 2-64 action to derive it.

Location code is commonly used to refer to the address site represented by the physical address. Using location codes eliminates the need to include the physical address in an XML message. For OAG messages, the location code is commonly found in the PARTNRID element of the PARTNER data type.

The Derive Address ID from Location Code action will access all trading partner locations across all organizations to derive the address ID. If the source location code is found in multiple organizations, the action will produce an error because it cannot uniquely identify an address ID.

The screenshot shows a configuration window for the 'Derive Address ID from Location Code' action. It contains four input fields with dropdown menus:

- 'Derive Target Address ID' with the value 'NAME'.
- 'Derive Target ORG ID' with an empty field.
- 'From Source Location Code' with the value 'PARTNRID'.
- 'with the Address Type of' with a dropdown menu showing 'Customer'.

See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Derive Target Address ID

Identify the target variable to store the address ID derived from the source location code. The value for the target variable is determined at runtime.

The target variable cannot be a global variable unless you are using it as a temporary variable to populate multiple application columns with the same value. The target variable cannot be a literal value.

### Derive Target ORG ID

Identify the target variable to store the organization ID derived from the source location code. The value for the target variable is determined at runtime.

The target variable cannot be a global variable unless you are using it as a temporary variable to populate multiple application columns with the same value. The target variable cannot be a literal value.

### From Source Location Code

Identify the source variable containing the location code. The source variable can be a global variable, whose value is determined at runtime, or a literal value.

### With the Address Type of

Select the address type from the list of values. The valid address types are as follows:

- Customer
- Supplier
- Internal Location
- Bank Branch

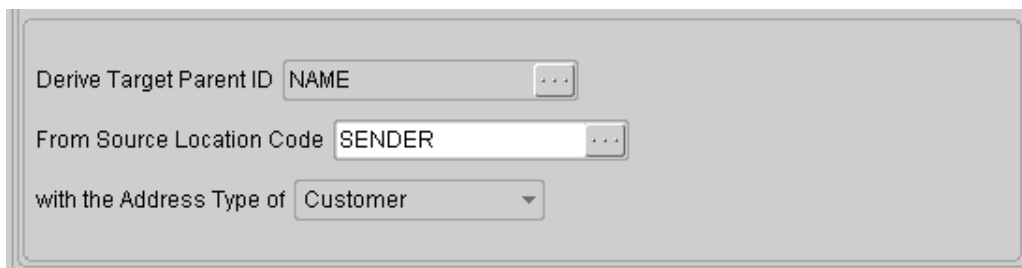
This address type is used by the XML Gateway Execution Engine to determine the appropriate Oracle address data model to access to derive the address ID.

## Map Action Editor - Derivations: Derive Parent ID from Location Code

The Derive Parent ID from Location Code action is used for inbound messages only. It derives the Parent ID associated with the site level source location code.

For example, given a SOLD-TO site in the XML message, this action will derive the parent customer associated with it.

Keep in mind that IDs are meaningful in the context of Oracle E-Business Suite only. The sender is not expected to know a valid parent ID.



The screenshot shows a configuration window for the 'Derive Parent ID from Location Code' action. It contains three fields: 'Derive Target Parent ID' with the value 'NAME', 'From Source Location Code' with the value 'SENDER', and 'with the Address Type of' with a dropdown menu set to 'Customer'.

See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Derive Target Parent ID

Identify the target variable to store the parent ID derived from the site level source location code. The value for the target variable is determined at runtime.

The target variable cannot be a global variable unless you are using it as a temporary variable to populate multiple application columns with the same value. The target variable cannot be a literal value.

### From Source Location Code

Identify the source variable containing the location code. The source variable can be a global variable, whose value is determined at runtime, or a literal value.

### With the Address Type of

Select the address type from the list of values. The valid address types are as follows:

- Customer
- Supplier
- Internal Location
- Bank Branch

This address type is used by the XML Gateway Execution Engine to determine the appropriate Oracle address data model to access to derive the address ID.

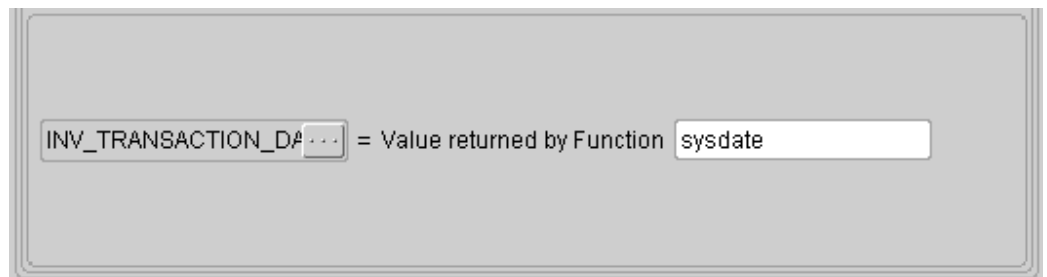
### Map Action Editor - Function Call: Execute Function Call

The Execute Function action calls a system function (for example, FND\_GLOBAL.USER\_ID) or any application function to perform an activity and return the result to the result variable.

Function calls with parameters are not supported by the Execute Function Call action.

See XML Gateway APIs, page F-1 for a list of special purpose functions.

**Note:** Procedure calls are not supported by the Execute Function Call action. See Map Action Editor - Procedure Call: Execute Procedure, page 2-78 for details on how to enable a procedure.



See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Result Variable

Select the source, target, or global variable to store the function return value.

### Value Returned by Function

Identify the name of the Function. The value returned by the function will be assigned to the result variable.

### Map Action Editor - Math Functions

Oracle XML Gateway supports the four basic math functions: addition, subtraction, multiplication, and division.

Each math function supports two operands. The Add, Subtract, Divide, and Multiply functions will each display the appropriate operators.

The image shows a graphical user interface for entering a mathematical expression. It features a label 'Tax Rate' followed by an equals sign (=), a text input field containing the value '8.25', a divide sign (/), and another text input field containing the value '100'. Each input field has a small square button with three dots to its right, likely for clearing or editing the field.

If the mathematical expression requires more than two operands, compute the expression two operands at a time, store the result in a temporary variable (source, target, or global), and then join the values in the temporary variables.

**Note:** Use the Function Call or Procedure Call action to utilize database level functions for complex mathematical computations.

See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

## Result Variable

Select the source, target, or global variable to store the result of the mathematical computation.

## Action Operands

Identify the two operands and the math function desired.

Each operand may be a variable (source, target, or global variables if defined) whose value is determined at runtime, or a numeric literal value.

## OAG Conversions

The availability of the OAG Conversion actions depends on the source and target. The following table summarizes when the actions are enabled:

Source	Target	OAG Actions
xml/dtd (OAG)	xml/dtd (OAG)	Both From and To actions are enabled.
xml/dtd (OAG)	database	Only From actions are enabled.
database	xml/dtd (OAG)	Only To actions are enabled.
xml/dtd (OAG)	xml/dtd (cXML)	Only From actions are enabled.
xml/dtd (cXML)	xml/dtd (OAG)	Only To actions are enabled.
xml/dtd (cXML)	database	None of the OAG actions are enabled.
database	xml/dtd (cXML)	None of the OAG actions are enabled.



## Map Action Editor - Convert to OAG DATETIME

The Convert To OAG DATETIME action converts the Oracle E-Business Suite representation for date to the OAG representation for date. This action is disabled if the target is database.

The OAG representation for date is as follows:

DATETIME (qualifier, type, index, YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, SUBSECOND, TIMEZONE)

The Convert to OAG DATETIME action converts the single column Oracle E-Business Suite date to the OAG DATETIME segment as follows:

- The DATETIME attributes for qualifier, type, and index must be set as default values in the Target Definition tab of the Transaction Map window.
- The DATETIME elements for YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, and SUBSECOND are derived based on the Oracle Application date value.
- The DATETIME element for TIMEZONE is determined as follows: The date and time data retrieved from the database, along with the value specified in the ECX: Server Time Zone profile option, are used to determine the Greenwich Mean Time (GMT) deviation. The deviation is used in the XML message generated by XML Gateway. No conversion is performed.

See XML Gateway Valid Time Zone Values, page E-1 for more information.

**Note:** The source and target elements are implicitly mapped via the Action. Additional element mapping will overwrite the original Action instructions.



See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Convert Source Datetime

Identify the source date/time element to be converted into the OAG DATETIME format. The source can be a literal value defined using a numeric format (such as 20020808 for 08-Aug-2002). Literal dates entered using a character format (such as 08-Aug-2002) are not valid.

Target and variables do not apply although they are available in the Element window.

The converted value will be assigned to the target DATETIME element selected when the action was invoked.

## Map Action Editor - Convert to OAG OPERAMT

The Convert To OAG OPERAMT action converts the Oracle E-Business Suite representation for operating amount to the OAG representation for operating amount.

The OAG representation for operating amount is as follows:

OPERAMT (qualifier, type, VALUE, NUMOFDEC, SIGN, CURRENCY, UOMVALUE, UOMNUMDEC, UOM)

The Convert to OAG OPERAMT action converts the single column Oracle E-Business Suite operating amount to the OAG OPERAMT segment as follows:

- The OPERAMT attributes for qualifier and type must be set as default values in the Target Definition tab of the Transaction Map window.
- The OPERAMT elements for VALUE, NUMOFDEC, SIGN, UOMVALUE, and UOMNUMDEC are derived based on the Oracle E-Business Suite operating amount value.
- The OPERAMT elements for CURRENCY and UOM are prompted for. If the appropriate Oracle E-Business Suite module columns are available, the values from the columns can be mapped to the CURRENCY and UOM elements.

**Note:** The source and target elements are implicitly mapped via the Action. Additional element mapping will overwrite the original Action instructions.



The screenshot shows a window titled 'Map Action Editor'. Inside, there is a section for 'Convert Source Operating Amount'. The 'Source Operating Amount' field is set to 'UNIT\_STANDARD\_PRICE' and the 'Target Operating Amount' field is set to 'OPERAMT'. Below these, the 'Source Currency' field is set to 'USD' and the 'Source Unit of Measure' field is set to 'EA'. Each field has a dropdown arrow to its right.

See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

## Convert Source Operating Amount

Identify the source operating amount element to be converted into the OAG OPERAMT format. The source can be a literal value or based on a global variable. The literal or variable must be numeric.

Target variables do not apply although they are available in the Element window.

The converted value is assigned to the target OPERAMT element selected when the Action was invoked.

## Source Currency

Identify the source currency code element if available or a literal value to be used by the Convert To OAG Action.

If an Oracle application column is available, the value from the column can be mapped to the CURRENCY element.

If no Oracle application column is available, the XML Gateway Execution Engine will leave the element blank.

Target and global variables do not apply although they are available in the Element window.

## Source Unit of Measure

Identify the source unit of measure code element if available or a literal value to be used by the Convert To OAG Action.

If an Oracle application column is available, the value from the column can be mapped to the UOM element.

If no Oracle application column is available, the XML Gateway Execution Engine will default the unit of measure code to EACH as recommended by OAG.

Target and global variables do not apply although they are available in the Element window.

## Map Action Editor - Convert to OAG QUANTITY

The Convert To OAG QUANTITY action converts the Oracle E-Business Suite representation for quantity to the OAG representation for quantity.

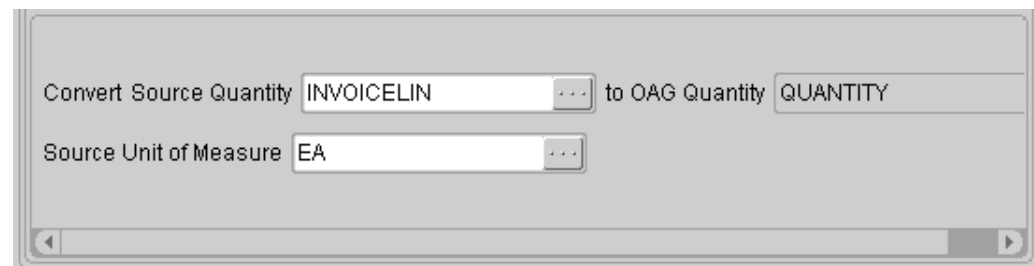
The OAG representation for quantity is as follows:

QUANTITY (qualifier, VALUE, NUMOFDEC, SIGN, UOM)

The Convert to OAG QUANTITY action converts the single column Oracle E-Business Suite quantity to the OAG QUANTITY segment as follows:

- The QUANTITY attribute for qualifier must be set as a default value in the Target Definition tab of the Transaction Map window.
- The QUANTITY elements for VALUE, NUMOFDEC, and SIGN are derived based on the Oracle E-Business Suite quantity value.
- The QUANTITY element for UOM is prompted for. If the appropriate Oracle E-Business Suite module column is available, the value from the column can be mapped to the UOM element.

**Note:** The source and target elements are implicitly mapped via the Action. Additional element mapping will overwrite the original Action instructions.



See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

## Convert Source Quantity

Identify the source quantity element to be converted into the OAG QUANTITY format. The source can be a literal (numeric) value or based on a global variable.

Target variables do not apply although they are available in the Element window.

The converted value is assigned to the target QUANTITY element selected when the action was invoked.

### Source Unit of Measure

Identify the source unit of measure code element if available, or a literal value to be used by the Convert To OAG Action.

If an Oracle application column is available, the value from the column can be mapped to the UOM element.

If no Oracle application column is available, the XML Gateway Execution Engine will leave the element blank.

Target and global variables do not apply although they are available in the Element window.

### Map Action Editor - Convert to OAG AMOUNT

The Convert To OAG AMOUNT action converts the Oracle E-Business Suite representation for amount to the OAG representation for amount.

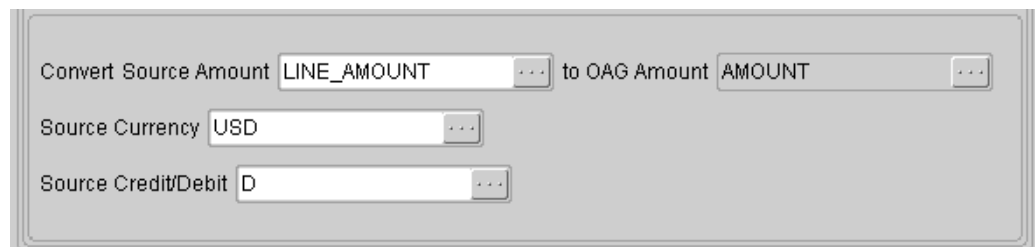
The OAG representation for amount is as follows:

AMOUNT (qualifier, type, index, VALUE, NUMOFDEC, SIGN, CURRENCY, DRCCR)

The Convert to OAG AMOUNT action converts the single column Oracle E-Business Suite amount to the OAG AMOUNT segment as follows:

- The AMOUNT attribute for qualifier, type, and index must be set as default values in the Target Definition tab of the Transaction Map window.
- The AMOUNT elements for VALUE, NUMOFDEC, and SIGN will be derived based on the Oracle E-Business Suite amount value.
- The AMOUNT element for CURRENCY and DRCCR are prompted for. If the appropriate Oracle E-Business Suite columns are available, the values from the columns can be mapped to the CURRENCY and DRCCR elements.

**Note:** The source and target elements are implicitly mapped via the Action. Additional element mapping will overwrite the original Action instructions.



See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Convert Source Amount

Identify the source amount element to be converted into the OAG AMOUNT format. The source can be a literal (numeric) value or based on a global variable.

Target variables do not apply although they are available in the Element window.

The converted value is assigned to the target AMOUNT element selected when the action was invoked.

### **Source Currency**

Identify the source currency code element if available or a literal value to be used by the Convert To OAG Action.

If an Oracle application column is available, the value from the column can be mapped to the CURRENCY element.

If no Oracle application column is available, the XML Gateway Execution Engine will leave the element blank.

Target and global variables do not apply although they are available in the Element window.

### **Source Credit/Debit**

Identify the source credit/debit flag if available or a literal value to be used by the Convert To OAG Action.

If an Oracle application column is available, the value from the column can be mapped to the DRCR element.

If no Oracle application column is available, the XML Gateway Execution Engine will derive the setting based on the amount value.

Target and global variables do not apply although they are available in the Element window.

## **Map Action Editor - Convert from OAG DATETIME**

The Convert From OAG DATETIME action converts the OAG representation for date to the Oracle E-Business Suite representation for date. This action is disabled if the source is database.

The OAG representation for date is as follows:

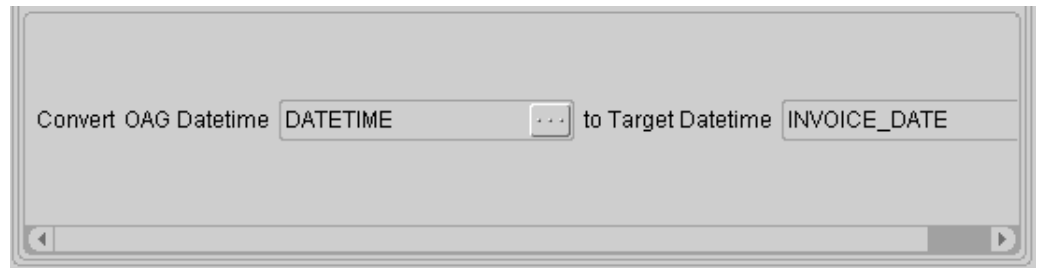
DATETIME (qualifier, type, index, YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, SUBSECOND, TIMEZONE)

The Convert From OAG DATETIME action converts the OAG DATETIME segment to the single Oracle E-Business Suite date column as follows:

- The DATETIME qualifier attribute can be used to determine the appropriate Oracle E-Business Suite date column to use.
- The DATETIME attributes for type and index are not used by Oracle E-Business Suite.
- The DATETIME elements for YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, and SUBSECOND are used to construct the Oracle E-Business Suite date value.
- The DATETIME element for TIMEZONE is determined as follows: if the time zone of the incoming message is different from the time zone specified in the profile option ECX: Server Time Zone, the incoming date and time are converted.

See XML Gateway Valid Time Zone Values, page E-1 for more information.

**Note:** The source and target elements are implicitly mapped via the Action. Additional element mapping will overwrite the original Action instructions.



See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Convert OAG Datetime

Identify the source DATETIME element to be converted into the Oracle E-Business Suite format. The source can be a literal value.

Target variables do not apply although they are available in the Element window.

The converted value is assigned to the application (target) date element selected when the action was invoked.

### Map Action Editor - Convert from OAG OPERAMT

The Convert From OAG OPERAMT action converts the OAG representation for operating amount to the Oracle E-Business Suite representation for operating amount.

The OAG representation for operating amount is as follows:

OPERAMT (qualifier, type, VALUE, NUMOFDEC, SIGN, CURRENCY, UOMVALUE, UOMNUMDEC, UOM)

The Convert From OAG OPERAMT action converts the OAG OPERAMT segment to the single Oracle E-Business Suite operating amount column as follows:

- The OPERAMT qualifier attribute can be used to determine the appropriate Oracle Application E-Business Suite module operating amount column to use.
- The OPERAMT type attribute is not used by Oracle E-Business Suite.
- The OPERAMT elements for VALUE, NUMOFDEC, SIGN, UOMVALUE, and UOMNUMDEC are used to construct the Oracle E-Business Suite operating amount value.
- The OPERAMT elements for CURRENCY and UOM can be stored in Oracle E-Business Suite if the appropriate columns are available.

**Note:** The source and target elements are implicitly mapped via the Action. Additional element mapping will overwrite the original Action instructions.

See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

## Convert OAG Operating Amount

Identify the source OPERAMT element to be converted into the Oracle E-Business Suite format. The source can be a literal value.

Target variables do not apply although they are available in the Element window.

The converted value is assigned to the application (target) operating amount element selected when the action was invoked.

## Target Currency

The source OPERAMT value is represented as a collection of subcomponents. Identify the application (target) element to store the currency code. Literal values are not allowed.

If an application (target) element is identified, the Convert From OAG Action will move the source currency code to it.

If no application (target) element is identified, the source currency code will be ignored.

## Target Unit of Measure

The source OPERAMT value is represented as a collection of subcomponents. Identify the application (target) element to store the unit of measure code. Literal values are not allowed.

If an application (target) element is identified, the Convert From OAG Action will move the source unit of measure code to it.

If no application (target) element is identified, the source unit of measure code will be ignored.

## Map Action Editor - Convert from OAG Quantity

The Convert From OAG QUANTITY action converts the OAG representation for quantity to the Oracle E-Business Suite representation for quantity.

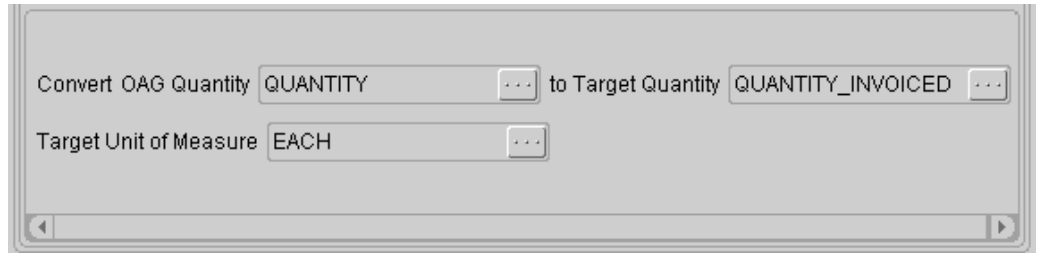
The OAG representation for quantity is as follows:

QUANTITY (qualifier, VALUE, NUMOFDEC, SIGN, UOM)

The Convert From OAG QUANTITY action converts the OAG QUANTITY segment to the single Oracle Application quantity column as follows:

- The QUANTITY qualifier attribute can be used to determine the appropriate Oracle E-Business Suite quantity column to use.
- The QUANTITY elements for VALUE, NUMOFDEC, and SIGN are used to construct the Oracle E-Business Suite quantity value.
- The QUANTITY element for UOM may be stored in Oracle E-Business Suite if an appropriate column is available.

**Note:** The source and target elements are implicitly mapped via the Action. Additional element mapping will overwrite the original Action instructions.



See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Convert OAG Quantity

Identify the source QUANTITY element to be converted into the Oracle E-Business Suite format.

Target variables do not apply although they are available in the Element window.

The converted value is assigned to the application (target) quantity element selected when the action was invoked.

### Target Unit of Measure

The source QUANTITY value is represented as a collection of subcomponents. Identify the application (target) element to store the unit of measure code. Literal values are not allowed.

If an application (target) element is identified, the Convert From OAG Action will move the source unit of measure code to it.

If no application (target) element is identified, the source unit of measure code will be ignored.

## Map Action Editor - Convert from OAG AMOUNT

The Convert From OAG AMOUNT action converts the OAG representation for amount to the Oracle E-Business Suite representation for amount.

The OAG representation for amount is as follows:

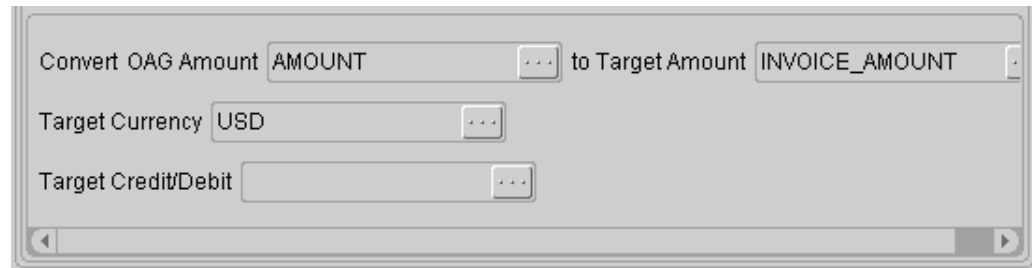
AMOUNT (qualifier, type, index, VALUE, NUMOFDEC, SIGN, CURRENCY, DRCR)

The Convert From OAG AMOUNT action converts the OAG AMOUNT segment to the single Oracle E-Business Suite amount column as follows:

- The AMOUNT qualifier attribute can be used to determine the appropriate Oracle E-Business Suite amount column to use.
- The AMOUNT attributes for type and index are not used by Oracle E-Business Suite.
- The AMOUNT elements for VALUE, NUMOFDEC, and SIGN are used to construct the Oracle E-Business Suite amount value.
- The AMOUNT elements for CURRENCY and DRCR can be stored in Oracle E-Business Suite if appropriate columns are available.



**Note:** The source and target elements are implicitly mapped via the Action. Additional element mapping will overwrite the original Action instructions.

The screenshot shows a window titled 'Map Action Editor' with a light gray background. It contains three rows of configuration options. The first row is 'Convert OAG Amount' followed by a text box containing 'AMOUNT' and a small square button with three dots. This is followed by 'to Target Amount' and a text box containing 'INVOICE\_AMOUNT'. The second row is 'Target Currency' followed by a text box containing 'USD' and a small square button with three dots. The third row is 'Target Credit/Debit' followed by an empty text box and a small square button with three dots. At the bottom of the window is a horizontal scrollbar.

See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Convert OAG Amount

Identify the source AMOUNT element to be converted into the Oracle E-Business Suite format. The source can be a literal value.

Target variables do not apply although they are available in the Element window.

The converted value is assigned to the application (target) amount element selected when the action was invoked.

### Target Currency

The source AMOUNT value is represented as a collection of subcomponents. Identify the application (target) element to store the currency code. Literal values are not allowed.

If an application (target) element is identified, the Convert From OAG Action will move the source currency code to it.

If no application (target) element is identified, the source currency code will be ignored.

### Target Credit/Debit

The source AMOUNT value is represented as a collection of subcomponents. Identify the application (target) element to store the credit/debit flag. Literal values are not allowed.

If an application (target) element is identified, the Convert From OAG Action will move the source credit/debit flag to it.

If no application (target) element is identified, the source credit/debit flag will be ignored.

### Map Action Editor - Other: Exit Program

The Exit Program action can be executed based on the result of executing a Procedure call, Function call, or retrieving the value of a Predefined Variable.

The Exit Program action rolls back the current transaction and exits the XML Gateway Execution Engine.

The Exit Program action has no additional operands.

See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

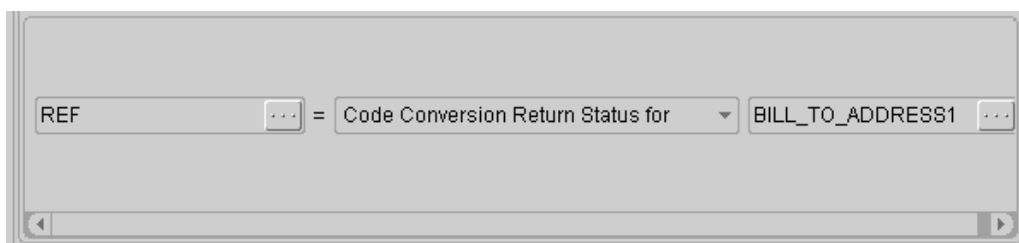
## Map Action Editor - Get Predefined Variable Value

The Get Predefined Variable Value action maintains predefined variables for the following:

- Inbound message envelope
- Code conversion status
- Transaction status

The values of these predefined variables are maintained by the XML Gateway Execution Engine and are available for inquiry and mapping to the Application Open Interface table column or Application API parameter.

You can control the process flow based on the values found in the predefined variables.



See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Result Variable

Select the source, target, or global variable to store the value of the predefined variable selected.

### Predefined Variables

Select the predefined variable whose value you are interested in. The variables exposed by the XML Gateway Execution Engine are as follows:

#### Code Conversion Return Status for

Select the source column enabled for code conversion. The code conversion process will search the trading partner-specific code conversions and then the standard-specific list, followed by the universal list.

The status of the code conversion is assigned to the result variable. The return status codes are shown in the following table:

Return Status Code	Description
0	Code conversion was successful.
1	<p>Code conversion process failed to find a match for the source column, the source value is copied to the target column.</p> <p>This is a warning only. It will not stop the XML Gateway Execution Engine. However, you may wish to take some action to control the process flow based on this status.</p> <p>The possible causes of this failure are: invalid code category passed to the code conversion API or code conversion cross-reference not found.</p> <p>Verify that the code category identified for the source column is valid (Refer to Seeded Code Categories, page B-1) and that a code conversion cross-reference is defined in the trading partner-specific list, the standard-specific list, or the universal list. Make the necessary corrections to prevent this error from occurring again.</p>
2	Code conversion process encountered unexpected error, session is terminated. Check the process log file for the details, make the necessary correction, and reprocess the message.

### Internal Control Number

The Internal Control Number is a system-generated number that uniquely identifies the XML message being processed. This number is useful for sending acknowledgments, for document audits, for document archival, and for troubleshooting.

The value for the Internal Control Number is assigned to the result variable identified for the action.

The Internal Control Number can be used to get envelope information. Refer to ECX\_TRADING\_PARTNER\_PVT.getEnvelopeInformation API, page F-14 for more information.

### Return Code

The Return Code is the error code associated with the error message. The error message may be returned to the trading partner contact or the XML Gateway system administrator using the Send Error Message action.

Multiple return codes can be concatenated in a single predefined variable. The value for the Return Code is assigned to the result variable.

The possible return codes are shown in the following table:

Return Code	Description
0	Success
1	Warning. The XML Gateway Execution Engine will not be stopped.
2	Failure. Check the process log file for the details, make the necessary corrections, and reprocess the message.

### Return Message

The Return Message is the text associated with the return code. The error message may be returned to the trading partner contact or the XML Gateway system administrator using the Send Error Message action.

Multiple return messages can be concatenated in a single predefined variable. The value for the Return Message is assigned to the result variable.

### **Receiver Trading Partner ID**

The Receiver Trading Partner ID is a unique identifier for the receiver of the XML message. The ID can be used to derive the trading partner name and other pertinent data.

Use the ECX\_TRADING\_PARTNER\_PVT.get\_receivers\_tp\_info procedure to derive the Party ID, Party Site ID, Organization ID, and Administrator e-mail Address associated with the Receiver Trading Partner ID. Refer to XML Gateway APIs, page F-1 for more information.

The Receiver Trading Partner ID is maintained for outbound messages and pass-through messages.

The value for the Receiver Trading Partner ID is assigned to the result variable.

### **Sender Trading Partner ID**

The Sender Trading Partner ID is a unique identifier for the sender of the XML message. The ID may be used to derive the trading partner name and other pertinent data.

Use the ECX\_TRADING\_PARTNER\_PVT.get\_senders\_tp\_info procedure to derive the Party ID, Party Site ID, Organization ID, and Administrator e-mail Address associated with the Sender Trading Partner ID. Refer to XML Gateway APIs, page F-1 for more information.

The Sender Trading Partner ID is maintained for inbound messages and pass-through messages.

The value for the Sender Trading Partner ID is assigned to the result variable.

### **Organization ID**

The Execution Engine derives the organization ID associated with the Sender or Receiver Trading Partner ID. The combination of the organization ID and the Sender or Receiver Trading Partner ID uniquely identifies a trading partner in the Oracle E-Business Suite.

## **Map Action Editor - Procedure Call: Execute Procedure**

The Execute Procedure action calls a procedure to perform an activity. Some procedures have input (send) parameters as well as output (return) parameters, while others have no parameters at all.

The Execute Procedure action connects to the database to provide a list of available procedures for selection. Once you select the procedure, you will be prompted for the parameter values.

Function calls are not supported by the Execute Procedure action.

See Message Designer APIs, page F-3 for a list of special purpose procedures for use with the Message Designer.

Procedure Name

See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

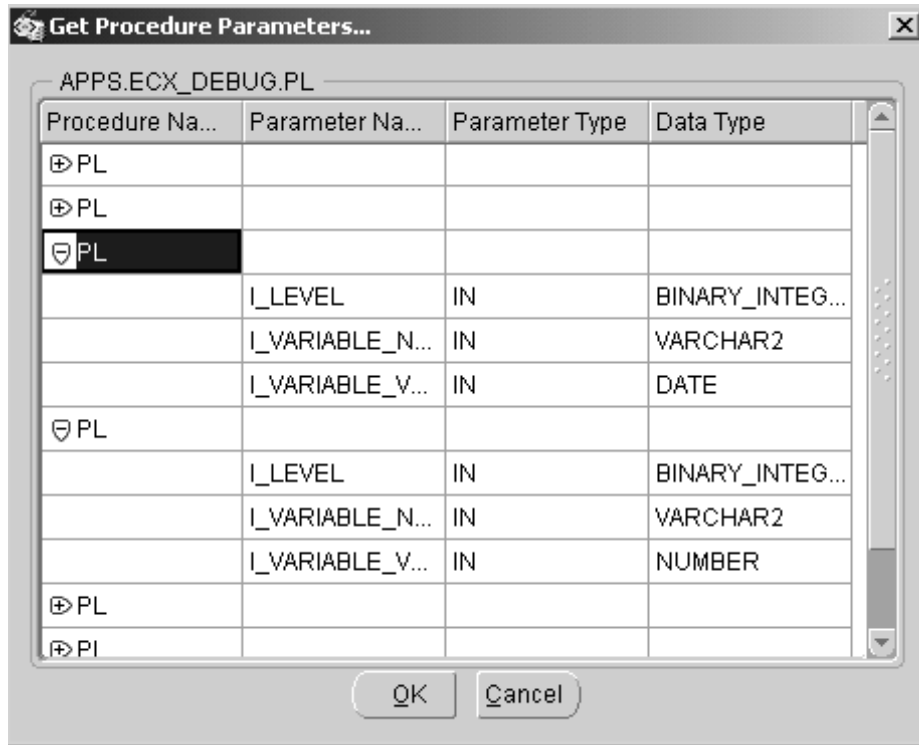
Click the Assign Parameter Value button to invoke the Get Procedure Parameters... window.

**Get Procedure Parameters...**

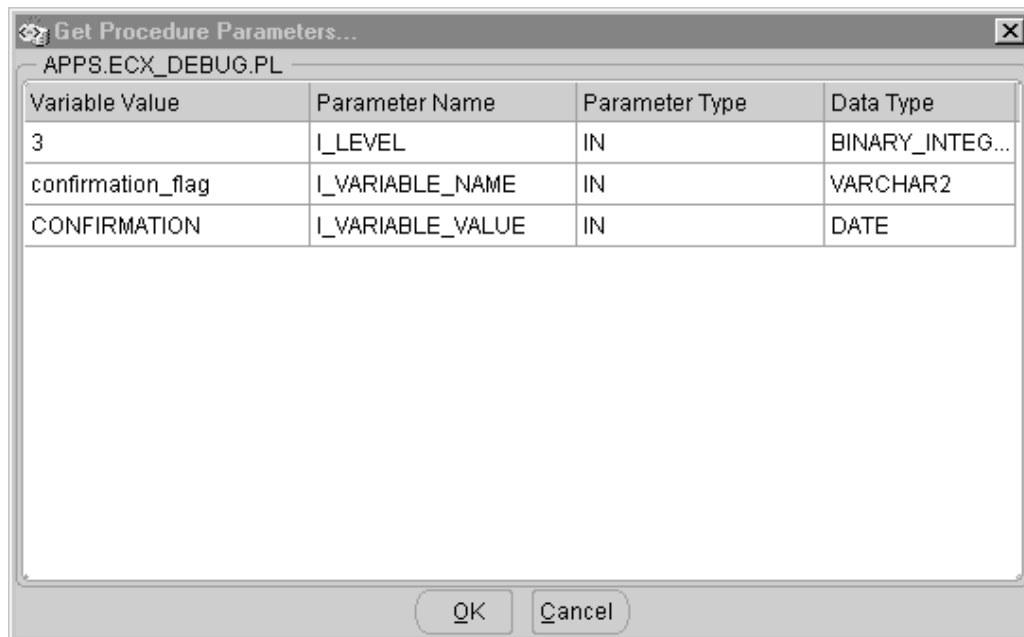
Username	<input type="text" value="appsecx"/>
Password	<input type="text" value="*****"/>
Connect to Database	<input type="text" value="ecxt"/>
Host	<input type="text" value="ec999sun"/>
Port	<input type="text" value="1533"/>
Schema Name	<input type="text" value="APPS"/>
Package Name	<input type="text" value="ECX_DEBUG"/>
Procedure Name	<input type="text" value="PL"/>

This window requests database connection information. The default database access information is displayed from the Database tab, page 2-5 of the File > Properties menu. The defaults can be overwritten on this screen. Changes made on this screen are not copied back to the Properties file. Changes are used for the current session only.

Enter the Schema Name, Package Name, and Procedure Name and click OK.



After connecting to the database, the window displays all signatures for the procedure. Expand all the signatures by clicking the "+". Select the procedure from the expanded list and click OK.



Enter the parameter values in this window. A parameter value may be a literal value, from a source or target, or a global variable.

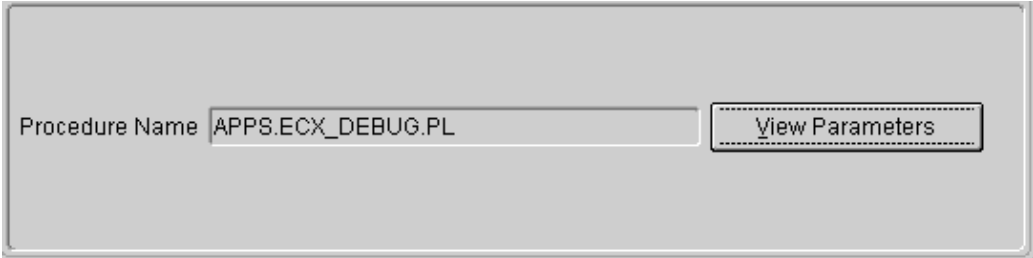
Regardless of how the parameters are defined in the procedure, a value is required (by the XML Gateway Execution Engine) for each parameter with a Parameter Type of IN unless a default value is provided by the procedure. The value can be based on a source variable, target variable, global variable (if defined), or a literal value. The value for the source, target, or global variable is determined at run time.

For procedures with return variables (Parameter Type of OUT), define temporary variables or use existing variables (source, target, or global) to store the return values. Status return values can be used to redirect the process flow based on severity. Return error codes and messages can be sent to the XML Gateway system administrator or Trading Partner contact using the Send Error Message action.

The XML Gateway Execution Engine supports parameters with a data type of VARCHAR2, DATE, NUMBER, CHAR, and CLOB. If the required procedure contains parameters of unsupported data types, the parameter values must be converted to a supported data type in order to utilize this action.

For parameters defined with default values, you can provide another variable or literal value if you wish to overwrite the default value.

Once you have entered all the required parameter values, click OK to return to the Map Action Editor window.

A screenshot of a software window titled "Map Action Editor". Inside the window, there is a label "Procedure Name" followed by a text input field containing the text "APPS.ECX\_DEBUG.PL". To the right of the input field is a button labeled "View Parameters". The window has a standard Windows-style border with a title bar and a drop shadow.

The procedure selected is displayed in the Procedure Name field. Click on the View Parameters button to review the parameter values entered. Verify the parameter values, make any necessary corrections, and click OK to return to the Map Action Editor window.

## Map Action Editor - Return Error Message: Send Error Message

The Send Error Message action sends an error message to either the Trading Partner contact or the XML Gateway system administrator identified in the ECX\_SYS\_ADMIN\_EMAIL system profile.

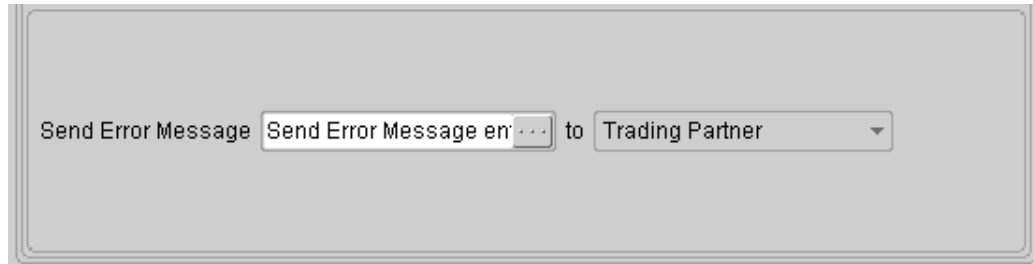
The Send Error Message Action is used for warnings that do not require the process to be terminated. For more serious errors requiring termination of the process, use the ECX\_ACTIONS.set\_error\_exit\_program API. See set\_error\_exit\_program, page F-19 for more information about its usage.

Error messages can be obtained from the following sources:

- Return parameter of a procedure call
- Status of a function call

A Workflow notification containing the error message is sent to the party identified in the To prompt.

This action is provided to augment the standard error handling and notification process.



See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Send Error Message

Identify the variable containing the error message to be sent. The error message can be represented as a literal.

When using this action in conjunction with the Procedure Call action, the variable may be a return parameter or a literal value.

When using this action in conjunction with the Function Call action, the variable may be the function return value or the value contained in the Return Message variable maintained by the XML Gateway Execution Engine.

You can concatenate several strings such as the value in Return Message with a literal value to form a complete message.

### To

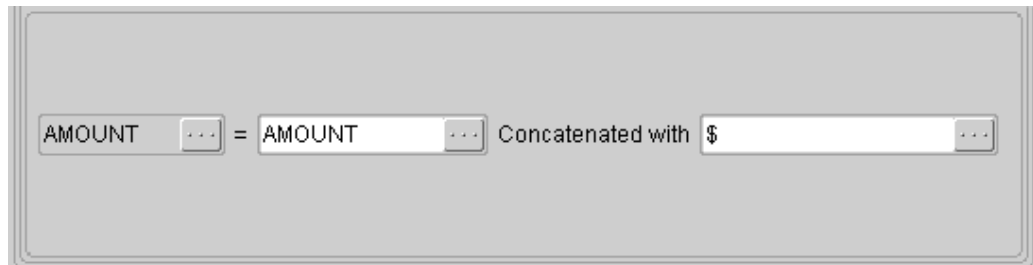
Identify the recipient of the error message. The valid options are the Trading Partner contact or the XML Gateway system administrator, identified in the ECX\_SYS\_ADMIN\_EMAIL system profile.

## Map Action Editor - String Functions: Perform Concatenation

The Perform Concatenation action concatenates the values in two operands. The operands can be based on variables whose values are determined at runtime, or based on literal values.

To concatenate more than two operands, start with the first two operands and store the result in a variable (source, target, or global). Then concatenate the value in the variable with the next operand.

The Perform Concatenation action can be applied to fields of any data type because the XML Gateway Execution Engine will convert the value to VARCHAR2 before performing the action.





See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Result Variable

Select the source, target, or global variable to store the resulting concatenated string.

### First Operand

Identify the first operand.

The operand can be a variable (source, target, or global variable if defined) whose value is determined at runtime, or a literal value.

### Concatenated With

Identify the second operand to be concatenated with the first operand.

The operand can be a variable (source, target, or global variable if defined) whose value is determined at runtime, or a literal value.

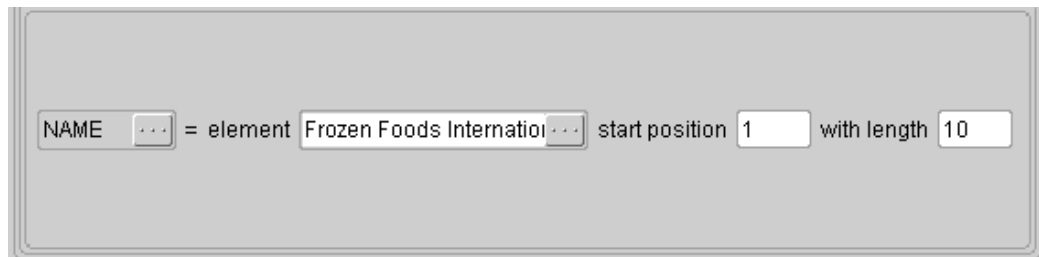
The concatenated value is assigned to the result variable.

## Map Action Editor - String Functions: Perform Substring

The Perform Substring action parses a given string from a start position and includes the characters within the length specified. The resulting substring is returned.

The Perform Substring action can be applied to fields of any data type because the XML Gateway Execution Engine will convert the value to VARCHAR2 before performing the action.

Refer to ECX\_CONDITIONS Package APIs, page F-11 for additional substring functions.



NAME ... = element Frozen Foods Internation ... start position 1 with length 10

See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

### Result Variable

Select the source, target, or global variable to store the resulting substring.

### Element

Identify the operand requiring substring action.

The operand can be a variable (source, target, or global variable if defined) whose value is determined at runtime, or a literal value.

## Start Position

Enter a start position greater than 0 but less than the maximum length of the Element.

## With Length

Enter a length greater than 0 but less than or equal to the maximum length of the Element minus the start position.

The substring function will process the element identified from the start position up to the length specified and assign the resulting substring to the result variable.

## Map Action Editor - XSLT Transformation

XML Gateway supports XSLT transformations for both inbound and outbound messages. An XSLT style sheet can be applied to an XML message before it is enqueued for delivery to a Trading Partner. Similarly for inbound messages, a stylesheet can be applied after the message is dequeued but before it is processed to the Oracle E-Business Suite.

The file containing the XSLT style sheet must be stored in the directory defined by the ECX\_UTL\_XSLT\_DIR system profile. The XML Parser applies the transformation as the last activity performed before an outbound message is sent, or as the first activity before an inbound message is processed.



See Map Action Editor - Overview, page 2-53 for details on the common components of the window.

## XSLT Style Sheet

The XSLT Stylesheet name can be a literal file name (such as ABC.xml) or a Global Variable (such as PARAMETER6).

## How to Extend DTDs

The XML Gateway Message Designer supports mapping to and from a DTD or an existing XML message. The use of an existing production XML message provides visibility to all the DTD extensions as well as duplicate nodes and elements necessary to support the actual business document. The use of a DTD is necessary if an existing XML message is not available. Using a DTD may require adding duplicate nodes, new segments, or elements to accommodate additional data.

When modifying an Oracle prebuilt message map or when creating a new message map that requires DTD extensions, follow the DTD extension guidelines outlined below. The Open Applications Group (OAG) has very strict guidelines regarding DTD extensions.

The only DTD extensions allowed are in the USERAREA. You can add elements in the appropriate OAG defined USERAREA or create new USERAREAs under the OAG defined USERAREA.

Refer to the OAG guidelines section 2.11 titled "USERAREA Extensions" in the *Open Application Group's Integration Specification* (available at <http://www.openapplications.org>).

Use USERAREAs to support descriptive flexfields as well as other extensions.

The OAG guidelines are as follows:

---

Top Level Name	<vendor.transaction.context.USERAREA>
Element Name	<vendor.elementname>
Reference File Name	<vendor prefix>_<product>_<version>.dtd

---

Oracle's adaptation of the OAG guidelines is as follows:

---

Top Level Name for Descriptive Flexfield Extensions	<Oracle.transaction.name of flexfield. USERAREA> Example: Oracle.ARInvoice.ARLINES.USERAREA
Element Name	<Oracle.attributen> where <i>n</i> is the number of attributes defined for the table
Top Level Name for other Extensions	<Oracle.transaction.product_prefix. USERAREA> Example: Oracle.ARInvoice.AR.USERAREA
Element Name	<Oracle.elementname>
Reference File Name	Oracle_<application shortname>DTDExtensions_<version>.dtd Example: Oracle_ARDTDExtensions_001.dtd

---

XML Gateway provides a template to be used as the basis to create the application-specific DTD extensions. The template contains a definition for the generic flexfields named "attribute\*" where \* is the number of attributes defined for the table. The application-specific DTD extension files are source-controlled with the specific application.

The application-specific DTD extension file can be updated as follows:

- Top level USERAREA names representing the application table containing the descriptive flexfields
- Descriptive flexfield names
- Top level USERAREA names for other extensions
- Element names for other extensions

The updates must be added after the existing USERAREA tag.

Use Conditional Node Mapping and key on Top Level USERAREA name to ensure that you are mapping from and to the correct group of flexfields or application extensions.

Be sure to remove extension definitions from the reference file if you are removing them from the message map.

Update the message map to incorporate the new fields introduced as a DTD extension. The updated message map and DTD extension files must be loaded in the XML Gateway repository for access by the XML Gateway Execution Engine. Refer to How to Load/Delete Message Maps and DTDs, page 2-87 for the details.

## How to Map a Pass-Through Message

Pass-through messages are messages that are received by Oracle Exchange for code conversion (if required) and then routed to a final destination. Oracle Exchange acts as a routing hub for these messages.

The map for a pass-through message is defined as follows:

**Note:** In a pass-through message, the same DTD is used for both the source and the target. If different DTDs are used the transaction is an XML-to-XML transformation, not a pass-through.

- Map the highest level only. The CNTROLAREA is the highest level in an OAG DTD.
- Load the message map into the XML Gateway repository for processing. Refer to How to Load/Delete Message Maps and DTDs, page 2-87 for details.
- Define the Exchange hub as the trading partner and enable the XML message as a pass-through transaction. See Trading Partner Setup, page 3-13.
- Define the necessary code conversions in Oracle XML Gateway or Oracle Exchange. See Code Conversion, page 3-26.

## How to Map to an API

This section describes how to map to an API instead of an Application Open Interface Table. Data for an inbound XML message is copied to a set of staging tables known as Application Open Interface tables. The data in these staging tables are validated using an Application Open Interface API containing the business rules. Valid data is moved to the Application base tables. Invalid data is marked for review.

In the case where the staging table and the validation rules are embedded in the same code (commonly referred to as the Application API), data for an inbound XML message is mapped to the parameters of the Application API.

To map directly to an Application API, the map must be defined as follows:

- The data definition type for the source and target must be XML.
- The source and target DTD and version must be the same.
- Map the source levels to the target levels.
- Define a target action for Execute Procedure referencing the Application API as the procedure name. The action is defined at the document level as a postprocess activity. Map the source variables to the API parameters. Include an action to check the API return status.

Apply this approach for each level of the document. If the document has three levels, this approach is applied three times.

- Load the message map and corresponding DTDs into the XML Gateway repository for processing. Refer to How to Load/Delete Message Maps and DTDs, page 2-87 to load the maps into the XML Gateway repository.
- Define the trading partner and enable the XML message.
- Define any necessary code conversions.

Message Designer does not support Application APIs consisting of multiple document levels. Message Designer is designed to process data one document level at a time.

## Loading and Deleting Message Maps and DTDs

### Loading/Deleting a Map

Oracle message maps are delivered and installed on the \$APPLTOP directory. They are automatically loaded into the XML Gateway repository using the LoadMap program. A map can be deleted using the DeleteMap program.

If you modified an Oracle prebuilt message map or created a new message map using the Message Designer, these maps are saved on your local file system. Perform the following steps to load the maps into the XML Gateway repository for use by the XML Gateway Execution Engine:

1. Move <mymap>.xgm file from the local file system to the middle tier. The apps.zip file containing the maps that were installed on the \$APPLTOP is installed on the middle tier.
2. Execute `java LoadMap<DB username><DB password><Hostname>:<Port>:<SID><mymap.xgm>` to load the message map into the XML Gateway repository. LoadMap will replace existing maps with the same name.

Example: `java LoadMap User1 welcome ap999sun.us.oracle.com:1521:ORA1151<mymap.xgm>`

**Note:** The LoadMap process will load a map if the map version is compatible with the engine version. The map version is stored in the <ECX\_MAJOR\_VERSION> and the <ECX\_MINOR\_VERSION> tags of the map file (.xgm). The engine version is stored in WF\_RESOURCES. Maps are compatible with the engine if the major version is the same and the minor version is the same or lower.

A map can be deleted using the DeleteMap program as follows:

1. Execute `java DeleteMap <DB username><DB password><Hostname><Port>:<SID><mapcode>` to delete a message map from the XML Gateway repository. LoadMap will replace existing maps with the same name, but obsolete maps with a different name must be deleted manually.

<mapcode> is the map name entered in the Message Designer.

Example: `java DeleteMap User1 welcome ap222sun.us.oracle.com:1521:ORA1151<mapcode>`

## Loading and Deleting a DTD

Each message map is associated with a set of DTDs: the main and its reference DTDs. Similar to message maps, the DTDs associated with the Oracle prebuilt message maps are delivered and installed on the \$APPLTOP. They are automatically loaded into the XML Gateway database using the LoadDTDToClob program. A DTD can be deleted using the DeleteDTDFromClob program.

To change a DTD referenced in an existing message map, use the File > Properties menu option of the Message Designer to make the necessary change. The updated message map must be loaded into the XML Gateway repository using LoadMap (refer to Loading/Deleting a Map, page 2-87). In addition, the new DTD used must be loaded into the XML Gateway database as follows:

1. Move `<mydtd>.dtd` file from the local file system to the middle tier. The `apps.zip` file containing the DTDs that were installed on the \$APPLTOP is on the middle tier.
2. Execute `java LoadDTDToClob<DB username><DB password><Hostname>:<Port>:<SID><mydtd.dtd> <RootElementName><Location>` to load the DTD into the XML Gateway database. LoadDTDToClob will replace existing DTDs with the same name.

`<RootElementName>` is the XML Root Element entered in the Message Designer.

`<Location>` is the subdirectory name entered in the Specify XML File and Root Element window of the wizard.

A DTD can be deleted using the DeleteDTDFromClob program as follows:

Execute `java DeleteDTDFromClob <DB username> <DB password> <Hostname>:<Port>:<SID><mydtd.dtd> <RootElementName><Location>` to delete a DTD from the XML Gateway database. LoadDTDToClob will replace existing DTDs with the same name, but obsolete DTDs with a different name must be manually deleted.

`<RootElementName>` is the XML Root Element entered in the Message Designer.

`<Location>` is the subdirectory name entered in the Specify XML File and Root Element window of the wizard.

LoadMap and LoadDTDToClob are provided as two separate programs to support various combinations of changes. To ensure that the maps and DTDs are synchronized, always execute LoadMap and LoadDTDToClob as a pair.

## Downloading a Map

You can download a map from the ECX table to produce the corresponding XGM file. Use the `DownloadMap.java` utility to download the map to the log directory.

The usage is as follows:

```
java DownloadMap <DB username><DB password><Hostname>:<Port>:<SID><MAP_CODE>
```

Note that there will be differences between the original map and the downloaded map.

## Differences Between the Downloaded Map and the Original Map

The downloaded map and original map will differ as follows:

- If the original map has a code category that is not present in `ecx_xref_hdr`, the downloaded map will not show the code category.
- If the original map uses variables of data type other than VARCHAR2, CHAR, NUMBER, CLOB, or DATE, the download map will default it to VARCHAR2.
- If the original map was created with a Message Designer version other than 2.6.0.X, the DownloadMap utility will default it to 2.6.3.0.0.
- If the original map was created with a version of Message Designer that did not include the dbdrv hints, these hints will be present in the downloaded map.

## Loading and Deleting an XSLT Style Sheet

XSLT style sheets are used to transform an XML message for rendering. The style sheets are created using an XSL editor. The xsl file can be staged in the file system or loaded into the database.

**Note:** The XML parser does not support nested style sheets using the `xsl: import` function. Define individual style sheets instead.

The ECX:XSLT File Path profile option identifies the file system directory where XSLT style sheets are staged. The engine checks the database first for the stylesheet, then if necessary, checks the file system.

Style sheets can be used by XML Gateway in one of two ways:

1. To be applied to an XML message generated or received by Oracle XML Gateway. This is accomplished by defining the XSLT Transformation action in the message map to reference a valid style sheet. Refer to XSLT Transformation, page 2-84 for details on how to define this action.
2. Passed with a well-formed and valid XML message to the `ECX_STANDARD.perform_xslt_transformation` API to process and return a transformed XML message for further processing by the calling environment. Refer to `ECX_STANDARD.perform_xslt_transformation` API, page F-1 for details regarding this API.

Because the engine will look for the style sheet in the database or the file system, you have the option of either loading or staging the stylesheet. To prevent accidental overwrite, a style sheet should reside in only one place.

Style sheets can be loaded into the database as follows:

1. Move `<myxsl>.xsl` file from the local file system to the middle tier.
2. Execute `java LoadXSLTToClob <DB username> <DB password> <Hostname>:<Port>:<SID> <myxsl.xsl> <Application_code> <Version>`  
  
`<Application_code>` is the subdirectory where the XSLT style sheet is source-controlled (for example: `ar/xml/xslt`)

If `<Version>` is not specified, the file with the highest version that matches the file name, application code, and file type is loaded into the `ecx_files` table. If a matching style sheet is found in the `ecx_files` table, the row is updated. If the specified style sheet is not found in the `ecx_files` table, a new row is added to the table with version number 0.

Style sheets can be deleted from the database as follows:

```
Execute java DeleteXSLTToClob <DB username> <DB password>  
<Hostname>:<Port>:SID> <myxsl.xml> <Application_code> <Version>  
<Application Code> is the name of the subdirectory where the XSLT style sheet is  
source-controlled (for example: ar/xml/xslt).
```

If version is not specified, the file with the highest version that matches the file name, application code, and file type is deleted from the `ecx_files` table. Execute this program multiple times to delete multiple versions of a style sheet.

## How to Implement Attachments in XML Messages

This topic consists of the following sections:

- Attachments and Oracle E-Business Suite, page 2-90
- Attachments and OAG Standard, page 2-90
- Attachments and Oracle Transport Agent, page 2-91
- Attachments and Outbound Documents, page 2-91
- Enable Attachments for Unit Test (Outbound), page 2-92
- Attachments and Inbound Documents, page 2-92
- Enable Attachments for Unit Test (Inbound), page 2-93

### Attachments and Oracle E-Business Suite

The Oracle Foundation module offers the ability to define attachments. An attachment may be defined as short text, long text, long RAW, or BLOB data type. When the attachment is defined in FND, unique identifiers are provided so that the attachment may be correlated to the business document. Example uses of attachments are terms and conditions associated with a purchase order or images associated with catalog items.

### Attachments and OAG Standard

With OAG version 7.X, the `ATTCHREF` data type is used to associate large objects with a business document. Attachments are associated to an XML message by `FILENAME`. The `ATTCHREF` data type is defined as follows:

#### **FILENAME**

(Required) The name of a file for reference purposes.

For XML Gateway message maps, the `CID` is mapped to the `FILENAME` element.

#### **CMPRSNTYPE**

(Optional) Identifies the method used to compress or minimize the size, if a file is attached to the document. This enables the receiving application to process the file appropriately.

For Oracle Foundation module, the compression type is determined when the attachment is deposited.

#### **CMPRSNID**

(Optional) Identifies the name of the compressed file. This enables the receiving application to find the file within the compressed package.



For XML Gateway, the physical file is not passed. The CID identified in the message map is used by OTA to construct the message payload with its associated attachments.

**DATETIME**

(Optional) The creation date of the attachment.

**DESCRIPTN**

(Optional) A free-form description of the transaction or any portion of the transaction.

**FILETYPE**

(Optional) Identifies the application source or format of the data within the file. Examples are MS Excel, CSV, and AutoCAD.

**NOTES1- NOTES99**

(Optional)

**QUANTITY**

(Optional) File size of the attachment.

**TITLE**

(Optional) Describes the formal name of "title" of the item, person, or object.

**Note:** The URL option supported by OAG 7.3 is not supported by XML Gateway at this time.

The ATTCHREF data type is not required if the attachment is a text string. These attachments can be associated with an XML data type or element for text strings. Textual attachments will be in-line with the business document. Large attachments of the BLOB data type will be associated off-line to the business document using the ATTCHREF data type as the reference.

## Attachments and Oracle Transport Agent (OTA)

The OTA component of Oracle XML Gateway is used to deliver business documents containing large objects of type BLOB in a Multipurpose Internet Mail Extension (MIME) payload over HTTP or HTTPS to a Trading Partner. Similarly, inbound documents containing attachments may be sent by a Trading Partner's OTA servlet and received by the Oracle E-Business Suite user's OTA servlet.

**Note:** As of the 11.5.9 release, support for large object attachments is limited to Business-to-Business document exchanges with Trading Partners delivered/received using OTA.

OTA is responsible for the following:

- Construction of the outbound XML message with valid MIME message containing the message payload and associated attachments.
- Extraction of the message payload and associated attachments from the inbound MIME message.

## Attachments and Outbound Documents

For outbound documents, it is assumed that an attachment has been defined in the Oracle Foundation (FND) module and that it has been associated with a business document. A given business document can have any number of attachments defined at any level of the document.

To include an attachment in the generated XML message, the XML Gateway must be informed of the relationship between the attachment and the business document. This is accomplished by adding a Procedure Call action to the message map to call the register attachment API at every point in the business document where an attachment is identified.

The relationship data is maintained in the ECX\_ATTACHMENT\_MAPS table, which is used by OTA to construct the outbound XML message with a valid MIME message containing the message payload and associated attachments. The register\_attachment API returns a unique content ID (referred to as CID) that is mapped to the OAG ATTCHREF data type, FILENAME element, or other standards-specific equivalent.

See register\_attachment API, page F-20 for more information.

Because the attachment, with its unique identifiers defined for it in FND, has a direct relationship with the map for the business document, any changes to the attachment (such as deletion) or to the attachment identifiers must be reflected in the message map to maintain data integrity.

Attachment content changes are reflected in the next usage of the attachment on a business document. The change will not be retroactively applied to previously generated business documents.

## Enable Attachments for Unit Test for Outbound Documents

The following outlines the steps necessary to set up attachments for outbound documents:

- Define attachments in the Oracle Foundation module.
- Associate the attachment with the business document.
- Create/update the message map to call the register\_attachment API for every off-line attachment associated with the business document.
- Map the off-line attachment (that is, BLOB data type) CID to the ATTCHREF data type, FILENAME element, or other standards-specific equivalent.
- Map the in-line attachment (that is, SHORT\_TEXT, LONG\_TEXT, or LONG\_RAW data type) to the appropriate XML data type or element for text strings.
- Load the map and DTD to the XML Gateway repository.
- Define the Trading Partner, enable the transaction for the Trading Partner, and use protocol type OTAH-ATCH, OTAHS-ATCH, HTTP-ATCH, or HTTPS-ATCH. The OTAH-ATCH and OTAHS-ATCH protocol types are used when an OTA envelope is required. The HTTP-ATCH and HTTPS-ATCH protocol types are used when an OTA envelope is not required.

## Attachments and Inbound Documents

For inbound documents, OTA will extract the business document from the multi-part message and enqueue it onto the ECX\_INBOUND queue. It will also extract the associated attachments and deposit them into the FND module of the receiving instance. For each attachment deposited into the receiving instance, an entry is written to the ECX\_ATTACHMENT\_MAPS table to record the relationship between the attachment and the business document.

If the inbound business document contains a reference to an attachment via the OAG ATTCHREF data type, FILENAME element, or other standards-specific equivalent, and the Oracle E-Business Suite module is interested in the attachment, the attachment content can be retrieved from the FND module of the receiving instance. This is accomplished by adding a Procedure Call action to the message map at every point of the business document where an attachment of interest is identified. There is no requirement to retrieve every attachment associated with an inbound business document unless the application module is interested in them.

Because the BLOB data type is handled at the database layer, the Oracle E-Business Suite module must define an API that internally calls the XML Gateway retrieve\_attachment API. The BLOB attachment content is passed from the retrieve\_attachment API x\_file\_data OUT parameter to the IN parameter of the application API. The application module is responsible for storing the attachment content in the appropriate column of the application table.

The receiving Oracle E-Business Suite module may change the FND attributes associated with the attachment originally deposited by OTA. This is accomplished by calling the retrieve\_attachment API to retrieve the original attachment, and then calling the reconfig\_attachment API to reset the FND attributes for the attachment with the application-specific identifiers.

See retrieve\_attachment, page F-22 and reconfig\_attachment, page F-23 for more information about these APIs.

## **Enable Attachments for Unit Test for Inbound Documents**

The following outlines the steps necessary to set up attachments for inbound documents:

- Create an application-specific API which internally calls the retrieve\_attachment API. This API is responsible for storing the retrieved attachment in the application column.
- Create/update the message map to check for the attachment reference (that is, OAG ATTCHREF data type, FILENAME element) and if the reference is present, call the application-specific API to retrieve each off-line attachment associated with the business document that the Oracle E-Business Suite module is interested in.
- Optionally create/update the message map to call reconfig\_attachment API for previously retrieved attachments to reset the FND attributes.
- Load the map and DTD to the XML Gateway repository.
- Define the Trading Partner, enable the transaction for the Trading Partner, and use protocol type OTAH-ATCH, OTAHS-ATCH, HTTP-ATCH, or HTTPS-ATCH. The OTAH-ATCH and OTAHS-ATCH protocol types are used when an OTA envelope is required. The HTTP-ATCH and HTTPS-ATCH protocol types are used when an OTA envelope is not required.



---

## XML Gateway Setup

This chapter explains the implementation steps required to set up XML Gateway. Topics include descriptions of all required forms as well as conceptual discussions of setup options.

This chapter covers the following topics:

- Setup Overview
- Define System Profile Options
- Assign XML Gateway Responsibility
- Define UTL\_FILE\_DIR Parameter
- Hub Definitions Form
- Define XML Standards Form
- Define Transactions Form
- Define Lookup Values
- Trading Partner Setup
- Code Conversion
- Trading Partner Code Conversion Form

### Setup Overview

#### Implementation Checklist

There are ten setup steps for the XML Gateway as shown in the following table:

Step	Completed	Task
1		Define System Profile Options, page 3-2
2		Assign XML Gateway Responsibility, page 3-4
3		Define the utl_file_dir parameters, page 3-4 (performed by a DBA)
4		Define Hubs, page 3-5
5		Define XML Standards, page 3-6
6		Define Transactions, page 3-7
7		Define Lookup Values, page 3-12
8		Define Trading Partners, page 3-13
9		Set up Trading Partner Code Conversion, page 3-34
10		Define Standard Code Conversion, page 3-32

**Note:** Additional setup steps are required for Web services. See the Web Services chapter for details.

## XML Gateway Forms

There are seven forms in the XML Gateway that you will use to complete the setup steps:

- System Profile Options, page 3-2
- Hub Definitions, page 3-5
- Define XML Standards, page 3-6
- Define Transactions, page 3-7
- Oracle XML Gateway Lookups, page 3-12
- Define Trading Partner Setup, page 3-13
- Define Code Conversion, page 3-32

## Define System Profile Options

The XML Gateway uses profile options to define the following for your system:

- The directory path for XML messages and log files
- The directory path for XSLT style sheets
- The XML Gateway System Administrator's e-mail address
- The sender's information system
- The time zone for the database server
- The maximum size an outbound document may reach before validating whether parsing should continue
- The Trading Partner user security feature for inbound transactions

To define these profile options, you must sign on to Oracle Applications as the System Administrator responsibility. Navigate to the System Profile Values form using the path (N) Profile > System.

For additional information on setting profile options, see the *Oracle Applications System Administrator's Guide*.

The following table lists the XML Gateway system profile options:

Profile Option	Description	Required	Default Value
ECX: Log File Path	Log File Path where the XML messages and runtime log are stored	YES	None
ECX: XSLT File Path	XSLT Path where XSLT style sheets are stored	YES	None
ECX: System Administrator Email Address	XML Gateway System Administrator e-mail address	YES	None
ECX: OAG_LOGICALID	Identifier for Sender's Information System	NO	None
ECX: Server Time Zone	The time zone in which the database server is running. See Important, page 3-3 below.	YES	Null
ECX: Maximum XML Size	Specifies the maximum size of an outbound XML document (in characters) beyond which parsing is performed based on the value of ECX: XML Validate Flag. If the size is not set, the document will be parsed by default. See Note, page 3-3 below.	NO	2 MB
ECX: XML Validate Flag	Specifies whether an outbound document should continue to be parsed by the engine after the ECX: Maximum XML Size has been met. See Note, page 3-3 below.	NO	Y
Enable User Security	Controls whether an inbound transaction can be enabled based on the value set in the profile option and the association between a user and Trading Partner.	NO	NO

**Important:** The valid values for ECX: Server Time Zone are listed in Appendix E, page E-1. If this profile option is not set, the time zone ID will default to Greenwich Mean Time (GMT).

**Note:** Setting the ECX: XML Validate Flag value to anything other than "Y" will turn off parser validation for a document once the maximum size has been parsed. Turning off the validation avoids Out of Memory errors for large documents.

If validation is turned off, the XSLT Transformation action cannot be performed. For more information on the XSLT Transformation action, see `perform_xslt_transformation`, page F-1.

For outbound Web services this validation must be turned off. See Web Services Setup, page 8-4.

For more information on setting profile options, see the *Oracle Applications System Administrator's Guide*.

## Assign XML Gateway Responsibility

Use the System Administrator responsibility to assign the Oracle XML Gateway responsibility to a user to access the Oracle XML Gateway database and forms. Use standard procedures to assign the responsibility.

For additional information on defining responsibilities, see the *Oracle Applications System Administrator's Guide*.

## Define UTL\_FILE\_DIR Parameter

Perform this step in cooperation with a Database Administrator (DBA).

### Define UTL\_FILE\_DIR Parameter in the INIT.ORA File

To use Oracle XML Gateway, you must first create directories where the XML message process log and XSLT style sheets will be stored. Oracle XML Gateway uses the UTL\_FILE package to read and write to the server.

UTL\_FILE can only write to accessible directories. The directories are defined by the `utl_file_dir` parameter in the `init<SID>.ora` file. This file is usually found in the `$ORACLE_HOME/dbs` directory. Within this file, each accessible directory is indicated by a line such as

```
utl_file_dir=<directory_name>
```

The specification of *directory\_name* will vary, depending on the operating system. If the operating system is case-sensitive, then *directory\_name* is case sensitive.

The value for *directory\_name* must be a physical directory. It cannot be a variable, a logical, or an alias. In addition, the value for *directory\_name* must match the value defined in the Oracle XML Gateway profile for ECX\_UTL\_LOG\_DIR File Path (ECX: Log File Path) and ECX\_UTL\_XSLT\_DIR File Path (ECX: XSLT File Path).

Refer to Define Profile System Values, page 3-2 for details.

### Unix Operating System

The following is an example of an entry for a UNIX operating system:

```
utl_file_dir=/d1/XML/logs/d1/XML/xslt
```

In addition to this form of database security, operating system security must also be considered. The file I/O operations performed with UTL\_FILE will be done by the Oracle user (the Oracle user is the owner of the files that are used to run the database, and also the owner of the processes that make up a database instance). Consequently, the Oracle user has to have operating system privileges to read from and write to all of the accessible directories. If the Oracle user does not have privileges for an accessible directory, then any operations in that directory will be prohibited by the operating system.

To ensure that operating system security allows the Oracle user to create, delete, rename, read, and write files in the specified directories, the DBA must grant directory and file access privileges by issuing the `CHMOD 777` command at the operating system level. This is a UNIX example only, so use appropriate operating system commands for your environment.



The Oracle instance must be brought down and back up for the changes in the `init<SID>.ora` file to be effective.

## Hub Definitions Form

Before defining a hub, you must first complete these setup steps:

1. Define System Profile Values, page 3-2
2. Define XML Gateway Responsibility, page 3-4
3. Define the `utl_file_dir` parameters, page 3-4 (performed by a DBA)

Navigate to the Hub Definitions form from the XML Gateway Responsibility by selecting Setup > Define Hubs.

A hub is an integration point within your network (either your intranet or the internet). Hubs are typically used to route documents to and from trading partners. Oracle Exchange is an example of a hub. You may also decide to create a private hub within your intranet through which all your ERP systems communicate.

The Hub Definitions form is used to define the hub and the authorized users conducting business via the hub. The hub users entered in this form will appear on the Trading Partner Setup form, page 3-13.

### Name (Required)

Enter the Hub name.

### Protocol Type (Required)

Protocol Type is the communication protocol associated with the hub, such as SMTP or HTTP. Select a value from the seeded list of values. The description for the protocol type is displayed.

### Protocol Address

When protocol type is HTTP or HTTPS, protocol address is prompted.

Protocol Address is the complete URL (including service/servlet) where the Transport Agent, page 7-1 will attempt to post the XML Document.

If the Protocol type is SMTP, the protocol address is an e-mail address.

## Hub Users

### Username (Required)

Enter the user name of the trading partner conducting business via the hub.

### Password (Required)

Enter the password for this user. The encrypted password is stored in the database. The password is not echoed when it is entered. You will be prompted on the same field to confirm the password.

### Hub Entity Code (Required)

Enter the hub entity code for this user.

Hub Entity Code has the same function as the Source Trading Partner Location Code, page 3-18 in the Trading Partner Setup form, page 3-13. It is the code found in the XML envelope to identify the source of the message.

If you are sending and receiving messages from a trading partner, you will have a mixture of your location codes and your trading partner's location codes depending on the direction of the message. For example, if you are sending messages out, the Source Trading Partner Location Code is your location code because you are the source of the XML message. If you are receiving messages, the Source Trading Partner Location Code is your trading partner's location code because they are the source of the XML message.

This code is examined for inbound messages during Trading Partner validation. When placed on an outbound message, the recipient will validate it as a valid source location code.

The Hub Entity Code is placed in the PARTY ID field in the XML Gateway envelope, page 4-6.

## Define XML Standards Form

Before defining XML standards, you must first complete these setup steps:

1. Define System Profile Values, page 3-2
2. Define XML Gateway Responsibility, page 3-4
3. Define the `utl_file_dir` parameters, page 3-4 (performed by a DBA)

Navigate to the Define XML Standards form from the XML Gateway Responsibility by selecting Setup > Define XML Standards.

This form defines standards bodies for XML messages, such as OAG.

The Standard Code entries made in this form will appear in a list of values for the Standard Code field in the following forms:

- Define Transactions form, page 3-7
- Trading Partner Code Conversion form, page 3-34 (accessed via the Trading Partner Setup form)
- Standard Code Conversion form, page 3-32

The Define Transactions form identifies the creating organization of the DTD, and hence, the XML message structure. The values entered in the Define XML Standards form will be the list of values for the Standard Code field in the Define Transactions form.

In the Trading Partner Code Conversion form and the Standard Code Conversion form, this value is used as part of the search key to access the code conversion tables.

Standard Code will also be the default standard code used in Standard Code Conversion table searches, when the Standard Code is assigned to the entries in the Define Transactions form.

### Standard Code (Required)

This field is the name or code for the standard's body for the XML messages.

Only XML Standards are entered in this form with the exception of the value UNIVERSAL. The seeded value UNIVERSAL is needed for the Standard Code Conversion form to identify universally used codes, such as ISO codes.

## Standard Type

Standard Type is any appropriate value that you choose.

## Description

Enter a description of the standard.

## Define Transactions Form

Navigate to the Define Transactions form from the XML Gateway Responsibility by selecting Setup > Define Transactions.

Use the Define Transactions form to define the transactions that will be used by the XML Gateway Execution Engine. You will then associate these transactions with a trading partner in the Trading Partner Setup form.

The Define Transactions form provides the following:

- A cross-reference between the external transaction identifiers and the internal Oracle transaction identifiers.
- Identification of the queue from which to retrieve inbound messages

## Cross-Reference Transaction Identifiers

This form provides a cross-reference between internal Oracle transaction identifiers, represented by the Transaction Type and Transaction Subtype, and External Transaction Types and External Transaction Subtypes.

**Note:** The data element pair of External Transaction Type and External Transaction Subtype, and the data element pair of Transaction Type and Transaction Subtype are not a one-to-one cross-reference by similar data element names. It is the combination of each pair that is significant to identify the external representation of the XML message, or the internal representation to identify the transaction to the Oracle Application.

For example: The following is an inbound message with a Party Type of "CUSTOMER": OAG PROCESS\_INVOICE\_003. The message has the External Transaction Type "PROCESS" and External Transaction Subtype "INVOICE." The inbound direction is determined by the XML Gateway.

The External Transaction Type "PROCESS" and External Transaction Subtype "INVOICE" will be matched to Transaction Type and Transaction Subtype equal to AP and INI as they are defined in the Define Transactions form. For this trading partner, the message map defines on which tables in Oracle Payables to place the inbound data.

Another example is an outbound message with a Party Type of "SUPPLIER":

Invoice transaction from Oracle Receivables has the Transaction Type "AR" and the Transaction Subtype "INO." The outbound direction is determined by the XML Gateway.

This Transaction Type and Transaction Subtype will be matched to the External Transaction Type "INVOICE" and External Transaction Subtype "PROCESS" as defined in the Define Transactions form for the specified trading partner. For this trading partner, the message map identifies what data to extract from Oracle Receivables.

The Party Type, Transaction Type, and Transaction Subtype are key codes used by Oracle E-Business Suite to integrate with the Workflow Business Event System (BES) to trigger message creation or message consumption.

## **Queues**

Different queues can be defined for various transactions from the application, or for XML messages to be transmitted. The queue names are assigned to the transactions via this form.

See Message Queues, page 5-1.

## **Define Transactions Form Fields**

### **Party Type (Required)**

Party Type defines the type of trading partner, such as Supplier, Customer, Bank, or internal locations (such as warehouses). Select a value from the list of values.

### **Transaction Type (Required)**

Transaction Type is the product short name for the base Oracle Application associated with the transaction, such as "AR" for Oracle Receivables. Refer to Transaction Type and Transaction Subtype Naming Conventions, page 3-9.

If you are using OAG standards, refer to Setting VERB and NOUN in OAG Standards, page 3-11.

### **Transaction Subtype (Required)**

Transaction Subtype is a code for a particular transaction within the application specified by the Transaction Type. The last position of the code represents the direction of the transaction: "I" for inbound, "O" for outbound.

See Transaction Type and Transaction Subtype Naming Conventions, page 3-9.

The combination of the Transaction Type and the Transaction Subtype identifies an Oracle transaction with which to associate this message. This data will appear on the Trading Partner Setup form.

If you are using OAG standards, refer to Setting VERB and NOUN in OAG Standards, page 3-11.

### **Transaction Description**

Enter a description for the transaction.

### **Standard Code (Required)**

The XML standard to be used for this transaction. The Standard Codes are set up in the Define XML Standards form, page 3-6. Choose the code from the list of values.

### **Direction (Required)**

Direction indicates if the message is inbound or outbound. Select "IN" for inbound messages, or "OUT" for outbound messages from the list of values.

### **External Transaction Type (Required)**

External Transaction Type is the primary external identifier for the XML message.

The combination of the External Transaction Type and the External Transaction Subtype should cross-reference this message to the Oracle internal transaction identified by the Transaction Type and the Transaction Subtype.

#### **External Transaction Subtype (Required)**

External Transaction Subtype is the secondary external identifier for the XML message.

The combination of the External Transaction Type and the External Transaction Subtype should cross-reference this message to the Oracle internal transaction identified by the Transaction Type and the Transaction Subtype.

#### **Queue (Required for Inbound Messages)**

A queue is a table in a database where transactions are staged for processing. Default queues are defined during installation. Select a queue from the list of values.

The field is disabled for outbound messages.

See Message Queues, page 5-1.

**Note:** Only queues with a prefix of ECX will display in the list of values.

### **Transaction Type and Transaction Subtype Naming Conventions**

Naming conventions for Transaction Type and Transaction Subtype are necessary to facilitate the reading of audit trails and for troubleshooting across applications.

The Transaction Type and Transaction Subtype codes should not focus on naming conventions based on the standard XML message that is implemented. A given transaction from an application may be mapped to several XML standards based on the trading partner agreements, likewise for inbound messages.

The following naming conventions are recommended:

#### **Transaction Type**

The Transaction Type is the product short name for the base Oracle Application. Examples of Oracle Application product short names are shown in the following table:

<b>Transaction Type</b>	<b>Application</b>
AP	Payables
AR	Receivables
OM	Order Management
PO	Purchasing
SS	Supplier Scheduling
RLM	Release Management
CUSTOM	(for user-defined messages)

#### **Transaction Subtype**

The Transaction Subtype is a code for a particular transaction within the application specified by the Transaction Type. The last position of the subtype code represents the direction of the transaction: "I" for inbound, "O" for outbound.

The naming convention is XXXXI or XXXXO where XXXX is significant to the application and its function, such as invoicing to Receivables and Payables.

For custom transactions, add a prefix to distinguish custom transactions from the Oracle supplied transactions.

The table below lists common combinations of Transaction Type and Transaction Subtype:

<b>Transaction Type (Application)</b>	<b>Transaction Subtype (Transaction)</b>	<b>Transaction Type's Processing Application</b>	<b>Transaction Type's Processing Application (Transaction Code plus Direction)</b>
PO	POCO	Purchasing	Purchase Order Change, Outbound
PO	POO	Purchasing	Purchase Order, Outbound
PO	POAI	Purchasing	Purchase Order Acknowledgment, Inbound
OM	POI	Order Management	Purchase Order, Inbound
OM	POCI	Order Management	Purchase Order Change, Inbound
OM	POAO	Order Management	Purchase Order Acknowledgment, Outbound
OM	POCAO	Order Management	Purchase Order Change Acknowledgment, Outbound
AR	INO	Receivables	Invoices, Outbound
AP	INI	Payables	Invoices, Inbound

The actual transaction detail is therefore recognized by its unique combination of the Transaction Type and Transaction Subtype. The Transaction Subtype does not have to be unique across applications, but only within the application specified by the Transaction Type.

Consequently, both the Purchasing and the Order Management application can use the same Transaction Subtype (for example, POI) because they may both import some type of purchase order. For example, Order Management will load orders from its customers; Purchasing may load from another purchasing application.

#### **Additional Suffix on Transaction Subtype**

If different database views are used to extract different types of transaction data, such as a blanket order versus a standard order, or different XML messages are needed for transactions such as a blanket order versus a standard order, then you can create different message maps using the Message Designer. You may also need to define different Transaction Subtypes to distinguish between such transactions in the Transaction setup.

When appropriate, a suffix can be attached to the base Transaction Subtype following the XXXXI or XXXXO naming convention to further distinguish a transaction in the XML

Gateway. For queries in a form, it is recommended to keep the same base Transaction Subtype XXXXI or XXXXO, so the Transaction Subtypes will follow each other in queries.

The suffix naming convention is therefore XXXXI-yyyy and XXXXO-yyyy where yyyy is the suffix.

The following table illustrates the Transaction Subtype naming convention using a suffix:

Transaction Type (Application)	Transaction Subtype (Transaction)	Transaction Type's Processing Application	Transaction Type's Processing Application (Transaction Code plus Direction)
PO	POO-BLK	Purchasing	Blanket Purchase Order, Outbound
PO	POO-REL	Purchasing	Purchase Order Release, Outbound
PO	POO-STD	Purchasing	Standard Purchase Order, Outbound

For example in a Purchasing application, all types of outbound purchase orders can be initiated under the general subtype POO with qualifiers for the various types of purchase orders such as blanket, standard, and release. Their subtype codes can be recognized as POO-BLK, POO-STD, and POO-REL respectively by the process. Avoid using long suffixes such as -BLANKET, -STANDARD, and -RELEASE to keep the naming convention simple. This sample is for illustration only.

## Setting VERB and NOUN in OAG Standards

The following topic discusses sources of data for the OAG CNTROLAREA.

The VERB and NOUN are key values required in the OAG CNTROLAREA data type. XML Gateway provides a database view that provides key data from the transaction table, while the attribute values are default values from the DTD used to create the outbound message map.

The purpose of the database view is to populate the VERB and NOUN elements illustrated below.

### The Verb and Noun in OAG's CNTROLAREA

```
<NOUN value = "INVOICE"> INVOICE </NOUN>
```

```
<VERB value = "PROCESS"> PROCESS </VERB>
```

When the External Transaction Type and External Transaction Subtype are entered into the transaction table through the Define Transactions form, they can be mapped to the NOUN and VERB value fields respectively by using the database view ECX\_OAG\_CONTROLAREA\_TP\_V (formerly ECX\_OAG\_CONTROLAREA\_V). See Transaction Map - Element Mapping, page 2-48.

**Note:** The ECX\_OAG\_CONTROLAREA\_TP\_V view is an upgraded version of the ECX\_OAG\_CONTROLAREA\_V view. Oracle XML Gateway supports both versions of the database view. For a detailed description of the differences, see the Note, page 2-25, in the Message Designer chapter.

**Important:** Within the OAG CONTROLAREA, the sequence of fields is VERB then NOUN in the message. In the Define Transactions form, the sequence of fields is NOUN in the External Transaction Type, then VERB in the External Transaction Subtype. Be careful not to reverse the order of the data.

## Define Lookup Values

To navigate to the Oracle XML Gateway Lookups form from the XML Gateway Responsibility select Setup > Define Lookup Values.

The Oracle XML Gateway Lookups form allows both entry and display of seeded data. This is a standard Oracle Application Object Library form.

Once a Type is selected in the upper section, its seeded values are displayed in the Lookup Details section of the form.

### Type (Required)

Type is the key code for the element stored in the seeded table.

The XML Gateway seeded lookup types are listed in the following table:

Type	Description	Sample Values
COMM_METHOD	Communications Method	HTTP, HTTP-ATTCH, HTTP-OXTA, HTTP-WM, HTTP-PS, HTTPS-ATTCH, HTTPS-OMB, HTTPS-OXTA, HTTP-PS-WM, IAS, ITG03, SMTP, OTAH-ATCH, OTAHS-ATCH, NONE, SOAP, JMS
CONFIRMATION_CODE	Confirmation Code	0, 1, 2
DOCUMENT	Documents/Transactions	CBODI and CBODO are seeded by Oracle XML Gateway. CBODI is OAG's inbound confirmation. CBODO is OAG's outbound confirmation.
MESSAGE_STANDARD	XML Message Standard	OAG, ORCL, RN, UNIVERSAL, PESC
MESSAGE_TYPE	Message Type	XML, EDI, FF (for flat file)
PARTY_TYPE	Party Types	B (for bank), C (for customer), S (for supplier), I (for internal location), E for Exchange, CARRIER (for Carrier)
TRANSACTION_CODE	Transaction Code	CBOD for confirmation BOD is the XML Gateway seeded transaction code.

### User Name (Required)

User name is defined by the user when data is entered.



**Application (Required)**

The text name for the application responsible for this Type.

**Description**

Description of the Type.

**Access Level**

The Access Level restricts changes that are possible to a lookup type. The possible levels are:

- System - No changes to the lookup codes are allowed.
- Extensible - New lookup codes can be added. However, you cannot modify seeded lookup codes.
- User - You can change any lookup code.

**Lookup Details****Code (Required)**

The seeded code for the lookup.

**Meaning (Required)**

The meaning of the lookup code.

**Description**

Description of the lookup code.

**Tag**

Not used.

**From Effective Date**

The starting effective date for the lookup code.

**To Effective Date**

The expiration date for the lookup code. The ending date is optional.

**Enabled Check Box**

Check the box if the Code is enabled. Remove the check to disable the Code.

**Trading Partner Setup**

Navigate to the Define Trading Partner Setup form from the XML Gateway Responsibility by selecting Setup > Define Trading Partners.

The Trading Partner Setup form is used to:

- Enable messages for the trading partner by identifying the internal and external transaction type and transaction subtype codes, and the XML standard associated with the message.
- Access the Trading Partner User Setup form.

- Access the Trading Partner Code Conversion form.
- Select a message map for the trading partner.
- Identify the communications protocol and address for a message. Optionally, the user can be selected from a hub.

For example, if JMS messages are used in the transactions, then you must select JMS as the Protocol Type and appropriate JMS queues in the Protocol Address fields.

This is the component that will enable a message to be processed through the XML Gateway engine. In the XML Gateway, the term "Trading Partner" refers to an entity such as a customer, supplier, bank branch, or internal locations at a particular address with which you exchange messages. Since a given entity may have several locations, you must define one Trading Partner for each customer address, supplier site, or bank branch as required for processing transactions by the Oracle XML Gateway.

During message processing, Trading Partner data is used to:

- Link a particular address location in Oracle E-Business Suite to the Trading Partner definition in the Gateway.
- Provide a means of telling the Execution Engine which Trading Partner message map to use.
- Enable specific transactions for Trading Partners.
- Determine how to deliver the message.

**Note: Multi-Org Consideration** Trading Partner setup in XML Gateway is organization-dependent. The list of Trading Partners and Trading Partner sites displayed is limited to those trading partners defined to the organization of the logon responsibility.

This form defines the parameters for the Trading Partner setup. The setup includes the identification of the trading partner site, the messages enabled for that site, and the delivery mechanism.

The Trading Partner Setup form requires an entry for each Transaction Type and Transaction Subtype associated with this trading partner.

The Trading Partner Setup form includes the following data to define the Trading Partner:

### Trading Partner Type (Required)

Trading Partner Type defines the type of trading partner, such as Supplier, Customer, Bank or internal location. Once the Trading Partner Type is selected from the list of values, the Trading Partner Names and Trading Partner Sites associated with the Trading Partner Type are displayed in the Trading Partner Name and Trading Partner Site lists of values below.

### Trading Partner Name (Required)

Given the selection in the Trading Partner Type, the appropriate Trading Partner Names are displayed in the Trading Partner Name list of values. For example, if Partner Type is Customer, then customer names will be displayed. These Trading Partners are limited to those trading partners associated with the organization of your logon responsibility. Select the appropriate Trading Partner Name.

**Trading Partner Site (Required)**

Given the selection in the Trading Partner Name, the appropriate Trading Partner Sites are displayed in the list of values. Select the appropriate Trading Partner Site.

**Company Admin Email (Required)**

This is the e-mail address of the administration contact to receive e-mails regarding warnings and errors. These notifications may be initiated by Oracle Workflow or by an action defined in the message map using the Message Designer. Users should check the error log.

**Code Conversion Button**

Use the Code Conversion button to access the Trading Partner Code Conversion form

See: Code Conversion, page 3-26

Trading Partner Code Conversion Form, page 3-34

**Trading Partner Details**

The combination of Party Type (Trading Partner Type), Trading Partner Name, Trading Partner Site, Transaction Type, and Transaction Subtype uniquely identify an outbound transaction.

The combination of External Transaction Type, External Transaction Subtype, Standard Code, and Source Trading Partner Location Code uniquely identify an inbound transaction.

The Trading Partner Setup form includes the following detail for each message:

**Transaction Type (Required)**

Transaction Type is the standard product short code for the base Oracle Application. These values are defined in the Define Transactions form. The list of values will display the available combinations of Transaction Type, Transaction Subtype, Standard Code, External Transaction Type, External Transaction Subtype, and Direction. Select the desired combination.

These values are only used internally to the XML Gateway. Refer to the Define Transactions Form, page 3-7 for details.

**Transaction SubType**

Transaction subtype is a code for a particular transaction within the application specified by the Transaction Type. The last position represents the direction of the transaction: I for inbound, O for outbound.

The combination of Party Type (Trading Partner Type), Transaction Type, and Transaction Subtype should identify an Oracle transaction with which to associate this message. These values are defined in the Define Transactions form.

These values are only used internally to the XML Gateway. Refer to the Define Transactions Form, page 3-7 for details.

**Standard Code**

The Standard Code associated with the Transaction Type selected above is displayed.

Standard Codes are set up in the Define XML Standards form, page 3-6.

### **External Transaction Type**

The External Transaction Type associated with the Transaction Type selected above is displayed.

The External Transaction Type is the primary external identifier for the XML message. These values are defined in the Define Transactions form, page 3-7 and they are found in the XML Gateway envelope, page 4-6.

The combination of the External Transaction Type and the External Transaction Subtype should identify this external message to the Oracle E-Business Suite.

### **External Transaction Subtype**

The External Transaction Subtype associated with the Transaction Type selected above is displayed.

The External Transaction Subtype is the secondary identifier for the XML message. These values are defined in the Define Transactions form, page 3-7 and they are found in the XML Gateway envelope, page 4-6.

The combination of the External Transaction Type and the External Transaction Subtype should identify this external message to the Oracle E-Business Suite.

### **Direction**

The Direction associated with the Transaction Type selected above is displayed.

This code identifies the direction of the transaction. The value IN identifies an inbound message, and the value OUT identifies an outbound message.

### **Map (Required)**

(Message) Map is the name of the map created using Message Designer.

Select the appropriate map from the list of values.

The naming convention for message maps consists of the four components: Transaction Type, Transaction Subtype, Standard and Release, and Direction. For example: "PO\_POO\_OAG70\_OUT" is the outbound Oracle Purchasing Purchase Order in the OAG standard, release 7.0. The direction code OUT makes the message direction more apparent in any list.

Refer to Message Designer, page 2-19 for more details on the naming convention.

If the desired map does not appear in the list of values, then the map has not been loaded into the XML Gateway database. Refer to the How to Load Message Maps and DTDs, page 2-87.

### **Connection/Hub (Required for outbound messages only)**

A DIRECT connect and a hub are the methods by which the message can be communicated. The XML message can be sent directly to a trading partner, or sent to a trading partner via a hub. The hub will then communicate the message to the trading partner.

Select DIRECT to conduct business directly with a trading partner, or select a hub from the list of values. If a hub is selected, then select a trading partner from that hub.

See the Hub Definition form, page 3-5 to define hubs.

Requirements for the entries for Protocol Type, Username, Password, and Protocol Address depend on whether you select DIRECT or a hub, page 3-5.

If you select DIRECT, complete these fields as follows:

- **Protocol Type (Required)**

Each message enabled for a Trading Partner includes a protocol type (also known as communication method). The associated correlation ID is determined internally and is associated with the message enqueued onto the agent (queue). Listeners defined for the agent will dequeue messages based on the correlation ID defined for it. For example, Oracle Transport Agent will only dequeue messages with a correlation ID of OXTA.

The following table shows the Protocol Type, Correlation ID, and Message System.

Protocol Type	Correlation ID	Message System
NONE	N/A	Message disabled
HTTP-WM, HTTPS-WM	WebMethods	Third party system
HTTP, HTTP-OXTA, HTTPS, HTTPS-OXTA, SMTP	OXTA	Oracle Transport Agent without attachments
OTAH-ATCH, OTAHS-ATCH	OXTA	Oracle Transport Agent with attachment using standard OTA envelope
HTTP-ATCH, HTTPS-ATCH	OXTA	Oracle Transport Agent with attachment and no OTA envelope
ITG03	ITG03	Oracle iProcurement Connector
IAS	IAS	Oracle Integration Server
SOAP	N/A	Web Service Agent
JMS	N/A	JMS Provider

Select the protocol type from the list of values. This data is seeded by the XML Gateway.

Protocol type NONE will disable the outbound message for this trading partner.

- **Username**

Enter the destination Username used to log in to the receiving server for the server that is identified in the server address.

For protocol types HTTP and HTTPS, Username and Password are required fields.

- **Password**

Enter the Password for the destination Username. The password is not echoed when it is entered. You will be prompted on the same field to confirm the password.

For protocol types HTTP and HTTPS, Username and Password are required fields.

- **Protocol Address**

Protocol Address is the complete URL (including service/servlet) where the Transport Agent, page 7-1 will attempt to post the XML Document.

For protocol type SMTP, the Protocol Address is an e-mail address, and is required.

For protocol type JMS, the Protocol Address is a JMS queue.

If you select a Hub, complete these fields as follows:

- **Protocol Type**

Protocol Type defaults from the Hub definition.

- **Username**

Select the Username from the list of values supplied by the Hub definition. If the Protocol Type is SMTP, the Username is not required. For all other Protocol Types, this field is required.

- **Password**

This field defaults to the password that is associated with the Username selected. The username and password combination is supplied from the Hub definition. If the Protocol Type is SMTP, this field is not required.

- **Protocol Address**

Protocol Address defaults from the Hub definition.

#### **Source Trading Partner Location Code (Required)**

Source Trading Partner Location Code is the code found in the XML Gateway envelope, page 4-6 to identify the source of the message. This is the code for the source trading partner of the message.

If you are sending and receiving messages from a trading partner, you will have a mixture of your location codes and your trading partner's location codes depending on the direction of the message. For example, if you are sending messages out, the Source Trading Partner Location Code is your location code because you are the source of the XML message. If you are receiving messages, the Source Trading Partner Location Code is your trading partner's location code because they are the source of the XML message.

This code is examined for inbound messages for Trading Partner validation. When placed on an outbound message, the recipient will validate it as a valid source or sending location.

This field is placed in the PARTY SITE ID in the XML Gateway envelope, page 4-6.

See: XML Gateway Envelope, page 4-6.

#### **Destination Trading Partner Location Code**

There are two types of routing in the XML Gateway: Static Routing and Dynamic Routing. Dynamic Routing allows the message to be rerouted by the first recipient of the message to the final destination recipient. Refer to the Routing, page 3-19 field below for Static Routing.

Destination Trading Partner Location Code is the code for the final recipient of the XML message. This code is not needed by the XML Gateway creating this message, but needed by the hub or the first trading partner receiving the message to identify their final trading partner to receive the message. It is the hub's or the first trading partner's code for that final destination trading partner.

For outbound messages, the final intended recipient location code identified by the Destination Trading Partner Location Code will be placed in ATTRIBUTE3 in the XML Gateway envelope, page 4-6.

For inbound messages, this code is found in ATTRIBUTE3 in the XML Gateway envelope. Refer to Static and Dynamic Routing, page 3-20 for an illustration that explains how ATTRIBUTE3 is populated.

### **Document Confirmation**

Document Confirmation is the indicator for the confirmation level that this Trading Partner would like to send or receive a confirmation.

0 (Default value) means Never send a confirmation

1 means Send a confirmation only if there are errors

2 means Always send a confirmation

It defines the condition under which a confirmation XML message is generated or received. Outbound messages receive inbound confirmations. Inbound messages generate outbound confirmations.

### **Routing**

Use the Routing field to identify another trading partner to whom messages will be routed. You select a trading partner for outbound transactions from the list of values. The inbound message is forwarded as an outbound message to the trading partner identified in the Routing field.

The XML Gateway provides both Static Routing and Dynamic Routing. Routing is the address to route the outbound message to when using the Static Routing method. Refer to the Destination Trading Partner Location Code, page 3-18 field above for Dynamic Routing.

See: Static and Dynamic Routing, page 3-20 for more information.

## **Required Communications Data**

### **Required Communications for Outbound Messages**

Different data is required depending on whether DIRECT or a trading partner is selected from a hub.

If DIRECT is selected, the following data fields are entered by the user depending on the protocol type for outbound messages:

- Protocol Type (required)
- Protocol Address (required)
- User Name (depends on Protocol Type)
- Password (depends on Protocol Type)
- Source Trading Partner Location Code to identify the recipient of the message (required)

If the message is sent via a hub, select the hub, then select a Username from the presented list. The following data fields are retrieved for outbound messages:

- The following data fields are copied from the Username in the hub definition:
  - Protocol Type
  - Protocol Address

- Hub Entity Code (acts like the Source Trading Partner Location Code for DIRECT communication)
- Password (optional and depends on Protocol Type)

If the message will be rerouted to another trading partner by the hub using Dynamic routing, then the following data is needed. It will be placed in ATTRIBUTE3 in the XML Gateway envelope, page 4-6:

- Destination Trading Partner Location Code

See: XML Gateway Envelope, page 4-6.

## Required Communications Data for Inbound Messages

Inbound messages require the following:

- Source Trading Partner Location Code to identify the sender of the message

## Static and Dynamic Routing

Messages can be passed through a middle trading party by using the static routing or dynamic routing feature. This method is also called a pass-through.

The following figure shows the routing flow of messages using the dynamic or static routing feature. The process starts with the review of the Trading Partner detail associated with the inbound message.

If data is found in the ATTRIBUTE3 field of the XML Gateway envelope, then the transaction is processed under the rules of Dynamic Routing. First the message is processed according to the inbound message map for the trading partner, then the message is rerouted to another trading partner who is the final recipient of the message. The trading partner detail for the final recipient is found in the Trading Partner Detail as an outbound message for the trading partner identified in the ATTRIBUTE3 field.

If there is no data in the ATTRIBUTE3 field of the XML Gateway envelope and there is data in the Routing field in the Trading Partner Detail for the inbound message, then the transaction is processed under the rules of Static Routing.

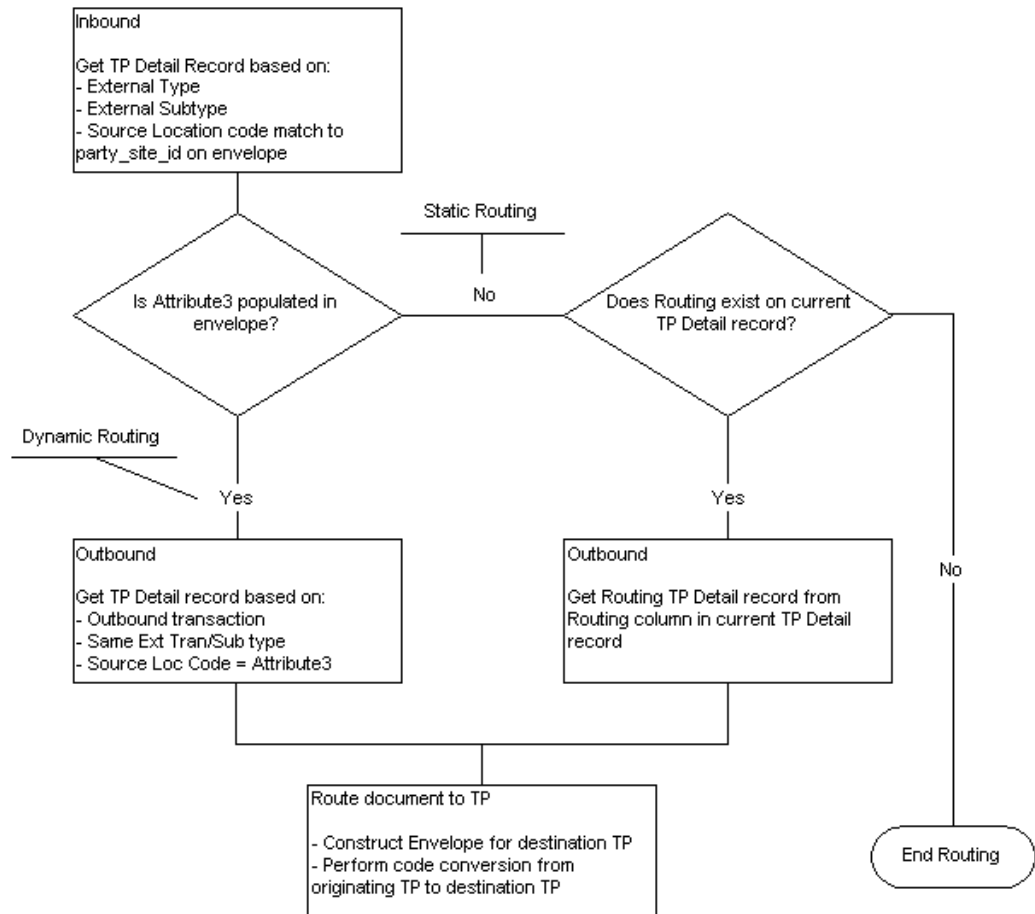
In Static Routing the message is processed according to the inbound message map for the trading partner, then the message is rerouted to another trading partner who is the final recipient of the message. The trading partner detail for the final recipient is found in the Trading Partner Detail as the trading partner identified in the Routing field for that inbound message.

If there is data in the ATTRIBUTE3 field of the XML Gateway envelope and there is data in the Routing field in the Trading Partner Detail for the inbound message, ATTRIBUTE3 will be used for the routing data.

If there is no data in the ATTRIBUTE3 field of the XML Gateway envelope and there is no data in the Routing field in the Trading Partner Detail for the inbound message, the message is not rerouted to another trading partner.

In both dynamic and static routing, data in the message may have code conversion applied at the trading partner level before constructing the outbound message for the final recipient. The retrieved code conversion values will be substituted into the XML message so the final trading partner receives the values that apply to them.





To explain Static and Dynamic routing for pass through messages, the following three trading partners are defined:

- Party 1: Initiator of the Message
- Party 2: First Recipient of the Message (This may be a hub)
- Party 3: Final Destination of the Message

Party 1 will send the message to Party 2. Then Party 2 will reroute it to Party 3.

## Static Routing

Static routing allows you to select another XML Gateway Trading Partner detail record from all your entries in the Trading Partner Setup, page 3-13 form. You can select a trading partner within the same trading partner or from any other displayed trading partner. Every time a message is received for that trading partner and that transaction, it will automatically be routed to the trading partner defined in the "Routing" column.

Static Routing occurs under the following conditions:

- The originator of the message (Party 1) does **not** include the final Destination Trading Partner Location Code in the message.
- For the given transaction and trading partner, the first recipient of the message (Party 2) has selected a trading partner in the Routing field to automatically forward that message to.

In order for this to happen, the following is necessary:

- Proper trading partner setup for the final recipient must be entered in the Trading Partner Setup in Party 2's environment.

### Static Routing Illustration

The following illustration reviews the required trading partner setup for each party in the example.

#### Party 1: Initiator of the Message:

1. In Static Routing, the first initiator of the message does *not* indicate the final Destination Trading Partner Location Code in their setup. No data should appear in ATTRIBUTE3 of the XML Gateway envelope. This is illustrated in the following table:

Entry	Trading Partner	Direction	External Transaction Type	External Transaction Subtype	Protocol	Source TP Location Code	Routing	Destination TP Location Code (in ATTR IBUTE3)
(1)	Party 2	OUT	INVOICE	ADD	HTTP	Party 1	N/A	N/A

#### Party 2: First Recipient of the Message:

PARTY 2 must define the message for both the inbound trading partner and the outbound trading partner to whom the message will be rerouted.

(2) Party 2 received the message from Party 1. The message was an inbound invoice to Party 2. See entry (2) of the table below.

(3) Party 2 will reroute the message to Party 3, because the trading partner setup for Party 3 was stored under the Routing field for Party 2's trading partner setup for Party 1. The trading partner setup data for the trading partner in the Routing field is used to create the new message and the XML Gateway envelope. See entry (3) of the table below.

Entry	Trading Partner	Direction	External Transaction Type	External Transaction Subtype	Protocol	Source TP Location Code	Routing	Destination TP Location Code
(2)	Party 1	IN	INVOICE	ADD	N/A	Party 1	(points to Party 3 for the outbound message)	N/A
(3)	Party 3	OUT (will be set to out)	INVOICE (use the same External Transaction Type)	ADD (use the same External Transaction Subtype)	HTTP	Party 2	N/A	N/A

## Dynamic Routing

The following method is dynamic because the first party originating the message indicates the Destination Trading Partner Location Code for the final destination trading partner to receive the message.

Dynamic Routing occurs under the following conditions:

- The originator of the message (Party 1) includes the final Destination Trading Partner Location Code in ATTRIBUTE3 in the XML Gateway envelope.
- The first recipient of the message (Party 2) can identify the trading partner code in ATTRIBUTE3 in their own Trading Partner setup.

In order for this to happen the following is necessary:

- Party 1's XML process must be set up to write the final trading partner to ATTRIBUTE3 of the XML Gateway envelope. The XML Gateway uses the Destination Trading Partner Location Code field in the Trading Partner Setup form.
- The proper trading partner setup must be entered in Party 2's process to act on ATTRIBUTE3. For the XML Gateway, this is in the Source Trading Partner Location Code field in the Trading Partner Setup form.

## Dynamic Routing Illustration

The following illustration reviews the required trading partner setup for each party in the example.

### Party 1: Initiator of the Message:

(1) In Dynamic Routing, the first initiator of the message (Party 1) passes a Destination Trading Partner Code in ATTRIBUTE 3 to the first message recipient (Party 2).

This code is defined by the first recipient of the message, so they can identify the trading partner to whom the message will be forwarded. The Destination Trading Partner Code will be copied to ATTRUBUTE3 in the XML Gateway envelope. See the table below:

Entry	Trading Partner	Direction	External Transaction Type	External Transaction Subtype	Protocol	Source TP Location Code	Routing	Destination TP Location Code (in ATTRIBUTE3)
(1)	Party 2	OUT	INVOICE	ADD	HTTP	Party 1	N/A	Party 3

### Party 2: First Recipient of the Message:

This second party must define the message for both the inbound trading partner and the outbound trading partner.

(2) Party 2 received the message from Party 1. The message was an inbound invoice to Party 2. See entry (2) in the Trading Partner Setup for Party 2 table.

Party 2 will validate Party 1 as a trading partner for that inbound message. The data is shown in the following table

Direction	External Transaction Type	External Transaction Subtype
IN	INVOICE	ADD

(3) Party 2 will reroute the message to Party 3. The message becomes an outbound invoice from Party 2. See entry (3) below.

Party 2 will also perform a trading partner lookup for the trading partner to whom the message is to be forwarded. The direction is changed to OUT. External Transaction Type and External Transaction subtype are copied from the inbound message. The lookup involves the data shown in the following table:

Direction	External Transaction Type	External Transaction Subtype
OUT	INVOICE	ADD

The Trading Partner Setup for Party 2, is shown in the following table:

Entry	Trading Partner	Direction	External Transaction Type	External Transaction Subtype	Protocol	Source TP Location Code	Routing	Destination TP Location Code (in ATTR IBUTE3)
(2)	Party 1	IN	INVOICE	ADD	N/A	Party 1	N/A	N/A
(3)	Party 3	OUT (set to OUT)	INVOICE (use the same External Transaction Type)	ADD (Use the same External Transaction Subtype)	HTTP	Party 2	N/A	N/A

### Party 3: Final Recipient of the Message:

(4) Party 3 received the message from Party 2. The message was an inbound invoice. The data is shown in the following table:

Entry	Trading Partner	Direction	External Transaction Type	External Transaction Subtype	Protocol	Source TP Location Code	Routing	Destination TP Location Code (in ATTR IBUTE3)
(4)	Party 3	IN	INVOICE	ADD	N/A	Party 2	N/A	N/A

## Trading Partner User Security

To ensure that only an authorized user is allowed to perform inbound XML transactions on behalf of any Trading Partner, Oracle XML Gateway provides the Trading Partner user security feature through the association of authorized users with specific Trading Partners using the Trading Partner User Setup form.

**Important:** Each Trading Partner can have one or more authorized users associated with it, but each user can be authorized for only one Trading Partner.

### What's the Impact?

**Oracle Transaction Agent (OTA) and Web Services** use internal APIs to validate the Trading Partner and authorize a user for a Trading Partner to perform XML transactions on behalf of a Trading Partner.

**Existing Users** will have to authorize an Oracle Applications user by associating a user with a Trading Partner using the Trading Partner User Setup form to perform XML transactions on behalf of a Trading Partner.

For backward compatibility, Oracle XML Gateway uses the Enable User Security profile option to turn the user security feature on or off.

- If the value is set to "No" (default) which turns the security OFF, the transaction will be enabled regardless of the association between the user and the Trading Partner.
- If you change the value from "No" to "Yes" which turns the user security ON, Oracle XML Gateway will validate the Trading Partner user against the Trading Partner user setup data to determine if the inbound transaction can be enabled.
  - If the user is linked to the Trading Partner, then the transaction is enabled for the Trading Partner.
  - If the user is not linked to the Trading Partner, then the transaction is not enabled. An error message will be shown in the transition monitor indicating that the user is not enabled in the XML Gateway Server. Please check your setup.

### The Trading Partner User Setup Form

The Trading Partner User Setup form allows you to associate authorized users with a Trading Partner. The same user cannot be associated with more than one Trading Partner.

Navigate to this form by clicking the User Setup button on the Define Trading Partner Setup form (XML Gateway Responsibility > Setup > Define Trading Partners).

#### Trading Partner Header

The Trading Partner Type, Trading Partner Name, and Trading Partner Site values are entered in the Trading Partner Setup, page 3-13 form and displayed here as read-only fields. You can update Company Admin Email address as needed. The address of the administration contact is to receive e-mail notifications regarding warnings and errors.

**Note:** If the Trading Partner Header information is not saved in the Trading Partner Setup form before selecting the User Setup button, an error message will occur.

#### User Information

##### User Name

Select valid Oracle Application user names to be associated with the Trading Partner from the drop-down list. An error message appears if the selected user has been assigned to any other Trading Partner.

**Note:** To update a selected user name, you need to first delete the selected name and then add a new one selected from the drop-down list so that the new user is also a valid Oracle Application user.

### **User Description**

This field populates automatically once the User Name field is selected.

## **Code Conversion**

The Oracle XML Gateway code conversion function provides a method to cross-reference the codes defined in Oracle E-Business Suite to codes used by trading partners, the XML standard, or other standard codes in the transactions.

For example, assume that the ABC Corporation transmits a purchase order to the XYZ Corporation. The XYZ Corporation processes the incoming data using its own Oracle values (for example, unit of measure, currency, or freight carriers), but XYZ is required to return ABC's codes. In this way, the trading partner that created the original transaction receives response transactions that use their own values.

The Code Conversion features include the following:

1. Define XML standard code conversion values defined by the XML standard. These codes are used by all trading partners.
2. Define universally used code conversion values that are defined by other standard organizations such as ISO, X12, and EDIFACT. These codes are used by all trading partners.
3. Define trading partner-specific code conversion values. The trading partner-specific code may override an XML standard or the other universally used code conversion values.

## **Code Conversion Setup Across XML Gateway Forms**

In the Define XML Standards form, Standard Codes such as OAG are entered to represent a message standard. There is a special purpose Standard Code called "UNIVERSAL" that is described below. See: Define XML Standards Form, page 3-6.

Entries from the Define XML Standards form appear in a list of values in the Define Transactions form that defines transactions to the XML Gateway. In this form, one Standard Code is associated with each transaction listed under External Processes. See: Define Transactions Form, page 3-7.

The actual code conversion values are entered in the Standard Code Conversion form and the Trading Partner Code Conversion form. The code conversion form provides a one-to-one code conversion from the Oracle values to an external value. The external code may be the XML standard codes such as OAG's code, the universal code such as ISO codes, or trading partner-specific codes. These are discussed below.

## **Code Conversion Values Marked for the Trading Partner**

The trading partner code conversion values are accessed through the Trading Partner Setup form. Refer to Trading Partner Setup, page 3-13 for details. Press the Code Conversion button to access the Trading Partner Code Conversion form from the Trading Partner Setup form.

The values associated with the trading partner's code values are the first set of codes to be examined during code conversion.

Before making entries in the trading partner specific table, first update the Standard Code Conversion values if necessary. This is important because the entries in the

Standard Code Conversion tables are displayed and potentially overridden in the Trading Partner Code Conversion form.

The following table illustrates the linking of a Trading Partner's message maps to Code Conversion:

Trading Partner	Message Map	Code Conversion Values are created for the Trading Partner?	The Message Map has a Unit of Measurement (UOM) field?	Accessed Trading Partner Code Conversion Values when the Message is Processed
Acme Corp, Atlanta	PROCESS_PO	Yes, for Acme, Atlanta	Yes, So Category UOM is entered for Code Conversion on the field	For UOM Category Code, the Acme, Atlanta code conversion values are accessed for the message map PROCESS_PO
Acme Corp, Chicago	PROCESS_PO	Yes, for Acme, Chicago	Yes, So Category UOM is entered for Code Conversion on the field	For UOM Category Code, the Acme, Chicago code conversion values are accessed for the message map PROCESS_PO
Acme Corp, Atlanta	ADD_INVOICE	Yes, for Acme, Atlanta	Yes, So Category UOM is entered for Code Conversion on the field	For UOM Category Code, the Acme, Atlanta code conversion values are accessed for the message map ADD_INVOICE
Acme Corp, Chicago	ADD_INVOICE	Yes, for Acme, Chicago	Yes, So Category UOM is entered for Code Conversion on the field	For UOM Category Code, the Acme, Chicago code conversion values are accessed for the message map ADD_INVOICE

- Message Map is assigned to this trading partner site in the Trading Partner Setup form (under Trading Partner Details)
- Access Code Conversion through the Trading Partner Setup form
- Data field is assigned the Category Code when the message map is created in the Message Designer

## Duplicate Entries Not Found

Since duplicate Oracle Values cannot be entered into the code conversion tables, the trading partner must always use the same "To Trading Partner Value" for all the transactions that it has enabled. The following table illustrates this rule:

Oracle Value	Description	From Trading Partner Value	To Trading Partner Value
Each	Each	EA	EA
Piece	Piece	PC	PC
Each	Each	PC	PC

- **Oracle Value** - Cannot key on multiple occurrences of the value "EACH"
- **From Trading Partner Value** - Cannot key on multiple occurrences of the value "PC"
- **To Trading Partner Value** - Cannot key on multiple occurrences of the value "PC"

## Code Conversion Values Marked as a Standard Code

Given the Code Conversion setup discussed above, there is a standard code associated with each message. The standard code is the source organization for the code value, such as OAG for the given XML message to be processed.

The values associated with that standard organization's code values are the second set of codes to be examined during code conversion.

## Code Conversion Values Marked UNIVERSAL

All other standards except the one that defined the XML message are marked collectively as UNIVERSAL. The XML Gateway does not store the other code conversion values under each standard such as the ISO codes or the X12 codes. It does not know which order the user wishes to access those code sets. For example, should the ISO codes be accessed before the X12 Codes, or the reverse? Hence, they are stored under a generic name.

The values associated with the UNIVERSAL code values are the third set of codes to be examined during code conversion.

The table below displays samples of ISO Country Codes that can be used across all trading partners and entered only once in the Standard Code Conversion forms.

Standard Code	Oracle Value	Description	From Trading Partner Value	To Trading Partner Value
UNIVERSAL	United States	United States	US	US
UNIVERSAL	United Kingdom	United Kingdom	UK	UK

If there is a conflict between two codes that must be entered as UNIVERSAL, it may be resolved by entering the conflicting codes under the Trading Partner code conversion for each trading partner needing that code.

Only the conflicting or duplicate code values need to be entered under each trading partner. The entries that do not cause conflicts can be entered under UNIVERSAL at the same time.



## Code Categories

A Category Code is a label for a set of entries in the code conversion table. For example, CARRIER is the category code for code conversion values associated with the carrier.

During transaction processing, only the code conversion table entries with an assigned category code are accessed for the given data element.

See Seeded Code Categories, page B-1 for the seeded list.

## Accessing the Code Conversion Values

### Key Access

Outbound and inbound transactions use different keys in the code conversion tables to access the codes.

- For outbound transactions, "Oracle Value" and the Standard Code are keys to access this table to determine the "To Trading Partner Value" to write in the transaction.
- For inbound transactions, "From Trading Partner Value" and the Standard Code are keys to access the table to determine the "Oracle Value" to pass to Oracle Application tables.

### Order of Table Search

The following table search order is performed for all transactions until a code conversion value table entry is found:

1. Access the Trading Partner code conversion table.

If the code is not found in this table, perform a second search.

2. Access the Standard code conversion table using the XML message's Standard Code, such as OAG. This Standard Code is associated with the transaction in the transaction table as defined via the Define Transactions form.

If the code is not found in this table, perform a third search.

3. Access the Standard code conversion table using the Standard Code UNIVERSAL. The universal entries may represent ISO codes or other standards.
  - For inbound transactions: If the code is not found in the tables after the three searches described above, then the "From Trading Partner Value" is copied to the target field in the message map.
  - For outbound transactions: If the code is not found in the tables after the three searches described above, then the "Oracle Value" is copied to the target field in the message map.

**Important:** If a code conversion value is not found in the table, it is not an error. There may be cases where only select values for a data element need code conversion. To require all values to have a code conversion table entry may cause you to do an excessive number of code conversion entries that are not necessary.

The following table illustrates the order that the tables and Standard Code Values are searched:

Search Order	Code Conversion Form	STANDARD CODE	Purpose
1	Trading Partner Code Conversion	CUSTOM	Define the trading partner-specific code conversion values. This includes any overrides to any standard's or universal code conversion values that are displayed from the Standard Code Conversion form. The trading partner code conversion values are used by all transactions for that trading partner.
2	Standard Code Conversion	For example: OAG ROSETTANET (ROS)	Define standard code conversion values for a given XML standard that are used across all trading partners. These values can be overridden for a specific trading partner in the Trading Partner Code Conversion form. The Standard Code field is the default XML standard associated with the XML message.
3	Standard Code Conversion	UNIVERSAL	Define standard code conversion values that are used across all trading partners and not associated with an XML Standard. These values can be overridden for a specific trading partner in the Trading Partner Code Conversion form. Those entries accommodate universally used code lists such as ISO, X12, EDIFACT, that can be used by all XML Messages.

### Outbound Transaction Code Conversion Table Access

For outbound transactions, the "Oracle Value" and the Standard Code in the code conversion tables are keys to retrieve the "To Trading Partner Value" to place it in the transaction. First the trading partner codes are searched. If an entry is not found, then the Standard code conversion table is searched for the standard codes and the universal codes.

If the code is not found in the tables after the three searches described above, then the Oracle Value is copied to the target field in the message map. Otherwise, the "To Trading Partner Value" will be copied to the target field in the message map.

The following table illustrates the code conversion concepts for outbound transactions using the category code UOM. The Outbound Search Key is comprised of the values for Conversion Table, Standard Code, and Oracle Value. The To Trading Partner Value is the data retrieved.

Conversion Table	Standard Code	Oracle Value	Description	From Trading Partner Value	To Trading Partner Value
Trading Partner or Standard	Appropriate Code for the first, second, third search	Each	Each	(ignore the value here - not relevant for Outbound Transactions)	EA (derive this code to place in the transaction)

The following table illustrates a sample outbound transaction code conversion for the category code UOM. The Outbound Search Key is comprised of the values for

Conversion Table, Standard Code, and Oracle Value. The To Trading Partner Value is the data retrieved.

Conversion Table	Standard Code	Oracle Value	Description	From Trading Partner Value	To Trading Partner Value
Trading Partner		Box	Box	(ignore the value here)	BX
Trading Partner	CUSTOM	Each	Each	(ignore the value here)	EA
Standard	OAG	Box	Box	(ignore the value here)	BX
Standard	OAG	Each	Each	(ignore the value here)	EA
Standard	UNIVERSAL	Box	Box	(ignore the value here)	BX

### Inbound Transaction Code Conversion Table Access

For inbound transactions, the "From Trading Partner Value" and the Standard Code in the code conversion tables are keys to retrieve the "Oracle Value." First the trading partner codes are searched. If an entry is not found, the Standard code conversion table is searched for the standard codes and then the universal codes.

If the code is not found in the tables after the three searches described above, then the "From Trading Partner Value" is copied to the target field in the message map. Otherwise, the "Oracle Value" will be copied to the target field in the message map.

The following table illustrates the code conversion concepts for inbound transactions using the category code UOM. The Inbound Search Key-1 is comprised of the values for Conversion Table and Standard Code. The Oracle Value is the retrieved data. The From Trading Partner Value is the Inbound Search Key-2.

Conversion Table (Inbound Search Key-1)	Standard Code (Inbound Search Key-1)	Oracle Value (Retrieve)	Description	From Trading Partner Value (Inbound Search Key-2)	To Trading Partner Value
Trading Partner or Standard	Appropriate Code for the first, second, third search	Each (Derive this code to place in the Oracle table)	Each	EA	(Ignore the value here - Not relevant for Inbound Transactions)

The following table illustrates a sample inbound transaction code conversion for the category code UOM. The Inbound Search Key-1 is comprised of the values for Conversion Table and Standard Code. The Oracle Value is the data retrieved. The From Trading Partner Value is the Inbound Search Key-2.

Conversion Table (Inbound Search Key-1)	Standard Code (Inbound Search Key-1)	Oracle Value (Retrieve)	Description	From Trading Partner Value (Inbound Search Key-2)	To Trading Partner Value
Trading Partner		Box	Box	BX	(Ignore the value here)
Trading Partner	CUSTOM	Each	Each	EA	(Ignore the value here)
Standard	OAG	Box	Box	BX	(Ignore the value here)
Standard	OAG	Each	Each	EA	(Ignore the value here)
Standard	UNIVERSAL	Box	Box	BX	(Ignore the value here)
Standard	UNIVERSAL	Each	Each	EA	(Ignore the value here)

### One Internal Code to Multiple External Codes

You can use the Action Assign Variable Value in Message Designer to assign multiple external codes given the single Oracle internal code by using conditions on the internal code. You must update the message in Message Designer as new codes are needed.

For example, the single Oracle internal carrier code stored in the SHIP\_VIA column may need to cross-reference to two external codes: the carrier and the transportation mode. For example,

If SHIP\_VIA = 'TRUCK-LAND'

move 'TRUCK' to the CARRIER.

If SHIP\_VIA = 'TRUCK-LAND'

move 'L' to the TRANSPORTATION\_METHOD.

If SHIP\_VIA = 'TRUCK-AIR',

move 'TRUCK' to the CARRIER.

If SHIP\_VIA = 'TRUCK-AIR'

move 'A' to the TRANSPORTATION\_METHOD.

#### **Note: For Outbound Transactions:**

If you have the SHIP\_VIA source data written twice to the file, then code conversion could be performed to convert the two fields separately: Once to derive the carrier code and once to derive the transportation method.

### Standard Code Conversion Form

Navigate to the Standard Code Conversion form from the XML Gateway Responsibility by selecting Setup > Define Code Conversion.

The Standard Code Conversion form displays a list of code values that are used by all trading partners unless the codes are overridden by trading partner specific code

values. For a discussion of trading partner specific codes see Trading Partner Code Conversion Form, page 3-34.

To read more about Code Conversion features and concepts see Code Conversion, page 3-26.

### Category Code

Select a category code to view its list of values in the Category Values portion of the form. A category code cannot be added or deleted.

### Description

Description of the category code.

### Category Values

#### Standard Code (Required)

A Standard Code identifies one of the following:

- The source organization for the code value, such as OAG, for the given XML message to be processed.
- The word UNIVERSAL, the seeded code to identify any other organizations such as ISO, X12, EDIFACT.

**Note:** Only XML standards and the single non-XML standard UNIVERSAL should be defined in the Define Transactions form. The process would not recognize standard codes such as ISO or EDIFACT though they can be entered into this form.

Select a standard code from the list of values.

Use the Define XML Standards form, page 3-6 to add new standard codes as needed.

#### Oracle Value (Required)

The Oracle Value is a code defined in the Oracle E-Business Suite, regardless if it is an inbound or outbound transaction. The Oracle Value is case-sensitive.

#### Description

Description of the Oracle Value.

#### From Trading Partner Value (Used with Inbound Transactions)

The "From Trading Partner Value" is a code in the message that represents data from the trading partner's perspective. This code is found in the inbound source transaction.

#### To Trading Partner Value (Used with Outbound Transactions)

The "To Trading Partner Value" is a code to be written in the outbound message. It represents data that the trading partner is expecting to receive. You cannot enter data in this field. This code value is always equal to and copied from the "From Trading Partner Value."

#### Standard Check Box

The Standard check box is enabled for all entries made in the Standard Code Conversion form.

If a code conversion value table entry has the Standard check box checked, and the code conversion value is later overridden in the Trading Partner Code Conversion table, this check box is switched "off" (made blank) in the Trading Partner Code Conversion form.

The system controls setting this check box off and on.

**Data Seeded Check Box**

Data can be seeded into the database by the XML Gateway. Seeded data is indicated by a check mark in this check box. If data is marked as seeded, it cannot be modified in the forms.

**REVERT ALL Button**

The Revert All Button is disabled in this form. It is used in the Trading Partner Code Conversion form to revert all codes back to the standard code value found in the standard Code Conversion table up to the last saving of the entries.

All fields except the Standard check box, Data Seeded check box, and the To Trading Partner Value can be changed.

**REVERT Button**

The Revert Button is disabled in this form. It is used in the Trading Partner Code Conversion form to revert the codes back to the standard code value found in the standard Code Conversion table up to the last saving of the entries.

## Trading Partner Code Conversion Form

The Trading Partner Code Conversion form allows the entry and display of code values for a specific trading partner.

Navigate to this form by clicking the Code Conversion button on the Define Trading Partner Setup form (XML Gateway Responsibility > Setup > Define Trading Partners).

The Trading Partner Code Conversion form contains data from the following sources:

- All trading partner-specific code values entered in this form
- All codes entered in the Standard Code Conversion form. The Standard check box on the form indicates if the data was entered in the Standard Code Conversion form. The standard code conversion values can be changed.

To read more about Code Conversion features and concepts see Code Conversion, page 3-26.

**Category Code**

Select a category code to view its list of values in the Category Values portion of the form. A code category cannot be added or deleted.

**Description**

Description of the category code.

**Category Values**

The following fields are found in the Category Values region:

**Standard Code**

When adding Trading Partner-specific code conversion values, this field defaults to CUSTOM.

## Oracle Value

The Oracle Value is a code defined in the Oracle E-Business Suite, regardless if it is an inbound or outbound transaction. The Oracle Value is case-sensitive.

## Description

Description of the Oracle Value.

## From Trading Partner Value (Used with Inbound Transactions)

The "From Trading Partner Value" is a code in the message that represents data from the trading partner's perspective. This code is found in the inbound source transaction.

## To Trading Partner Value (Used with Outbound Transactions)

The "To Trading Partner Value" is a code to be written in the outbound message. It represents data that the trading partner is expecting to receive. You cannot enter data in this field. This code value is always equal to and copied from the "From Trading Partner Value."

## Standard Check Box

The Standard check box is enabled for all entries made in the Standard Code Conversion form.

If a code conversion value table entry has the Standard check box checked, and the code conversion value is later overridden in the Trading Partner Code Conversion table, this check box is switched "off" (made blank) in the Trading Partner Code Conversion form.

The system controls setting this check box off and on.

## Data Seeded Check Box

Data can be seeded into the database by the XML Gateway. Seeded data is indicated by a check mark in this check box. If data is marked as seeded, it cannot be modified in the forms.

## REVERT ALL Button

The Revert All Button in the Trading Partner Code Conversion form reverts all codes back to the standard code value found in the standard Code Conversion table up to the last saving of the entries.

All fields except the Standard check box, the Data Seeded check box, and the To Trading Partner Value can be changed.

## REVERT Button

The Revert Button in the Trading Partner Code Conversion form reverts the codes back to the standard code value found in the standard Code Conversion table up to the last saving of the entries.

**Note:** The Revert Button applies to only the selected entry for this specific trading partner.

## What Can Be Updated

If you are adding a Custom code, all fields except the Standard check box, and Data Seeded check box can be changed.

If the Standard Code is not "CUSTOM" then only the From Trading Partner Value and the To Trading Partner Value can be changed. The Standard check box will be enabled.



---

## Execution Engine

This chapter covers the following topics:

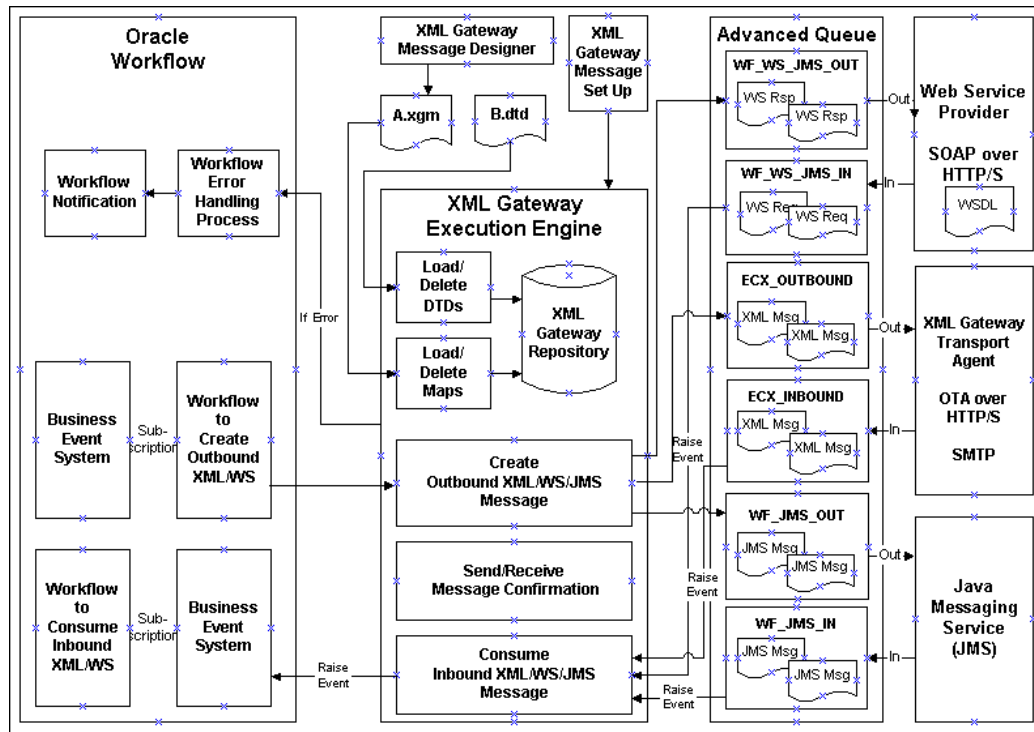
- Execution Engine Overview
- Protocol Type
- XML Gateway Envelope
- Trading Partner Validation for Inbound Messages
- Trading Partner Validation for Outbound Messages
- How to Implement the OAG Confirmation Business Object Document

### Execution Engine Overview

The XML Gateway consists of three components: the Message Designer, Setup, and the Execution Engine. It interfaces with the following Oracle products:

- Oracle Transport Agent for message delivery
- Oracle Advanced Queuing for message propagation, and queue management
- Oracle Workflow Business Event System to publish and subscribe to business events. Workflow also provides an active e-mail notification to report errors detected by the XML Gateway Execution Engine, Advanced Queuing (AQ), or the transport agent

The following diagram illustrates the process flow of XML messages through the XML Gateway Execution Engine using all the components mentioned above. Details of the XML Gateway Execution Engine are summarized below.



The XML Gateway Execution Engine can process messages properly after the following:

- Message maps are created and loaded into the repository along with their associated DTDs.
- Trading Partners are defined.
- Code Conversions are defined.
- Transactions are defined.
- Oracle Workflow Business Event System events are published by the Oracle E-Business Suite and subscriptions to those events are defined.
- Engine and listeners are started.

The XML Gateway listeners are actively polling for interested events. The Execution Engine will begin processing once Oracle Workflow Business Event System detects an outbound transaction to be processed, or that an inbound message has arrived on the queue.

The XML Gateway Execution Engine does the following during processing:

- **(Inbound Messages) Dequeue Message from Inbound Queue**
- **Validate Message via XML Parser**

(Inbound Message) Uses the XML Parser to validate the inbound message to determine if it is well-formed and valid (based on a DTD stored in the DTD directory) before proceeding further.

(Outbound Message) Uses the XML Parser to validate the newly created message to ensure that it is well-formed and valid. A poorly formed or invalid message

(based on a DTD stored in the DTD directory) will not be enqueued onto the Outbound Queue.

- **Validate Trading Partner or Hub**

If the inbound message is both well-formed and valid, the Execution Engine proceeds to validate that the Trading Partner and document are defined. If the Trading Partner is not defined or the document is not defined for the Trading Partner, the XML message will not be processed further.

- **Get Message Map from Repository**

If the message map associated with the Trading Partner is not available in the XML Gateway repository, the XML message will not be processed further.

- **Execute the Message Map**

- **(Outbound Messages) Gather Application Data**

If the Trading Partner is valid and the message map exists in the repository, the Execution Engine gathers the application data from the Oracle e-Business Suite using the database views and columns identified in the message map.

- **(Inbound Messages) Maps Data**

If the Trading Partner is valid and the message map exists in the repository, the Execution Engine maps the data in the XML message to its target data fields in Oracle e-Business Suite tables and columns identified in the message map. These are often the Application Open Interface tables.

- **Apply Code Conversion**

Apply code conversion for source columns enabled for code conversion.

- **Apply Actions**

Apply actions where defined (may be document, element, or root level).

(Inbound Message) Action codes include inserting data into the Application Open Interface tables, then executing the Application Open Interface API to populate the base application tables.

- **(Outbound Messages) Create XML Message**

Create XML message using the message map and the application data as described above.

- **Detect and Report Processing Errors**

Errors may be detected by the Oracle XML Gateway Execution Engine, Oracle Advanced Queuing, Oracle Workflow, or Oracle Transport Agent. Information regarding the error is enqueued onto the Error Queue. An e-mail notification is sent via Oracle Workflow to notify the trading partner regarding data errors, or the XML Gateway system administrator regarding system or process errors.

In addition, for system or process errors, a copy of the XML message is placed in the XML message directory for use in troubleshooting the reported error. For trading partner-related data errors, the trading partner can refer to the copy of the XML message.

- **Copy XML Message**

For system or process errors, a copy of the XML message is placed in the XML message directory for use in troubleshooting the reported error. For trading partner-related data errors, the trading partner can refer to their copy of the XML message.

- **(Outbound Messages) Enqueue Message to Outbound Queue**

Enqueue well-formed and valid message onto the Outbound Queue. Oracle Transport Agent will dequeue the message from the Outbound Queue and deliver it to the trading partner.

- **Create or Consume Confirmation Messages**

(Inbound Message) Receive confirmation message, if it is enabled for the Trading Partner and confirmation is requested on the outbound message, or a default is set up for the trading partner.

(Outbound Message) Create a confirmation message, if it is enabled for the Trading Partner and confirmation is requested on the inbound message, or a default is set up for the trading partner.

## Protocol Type

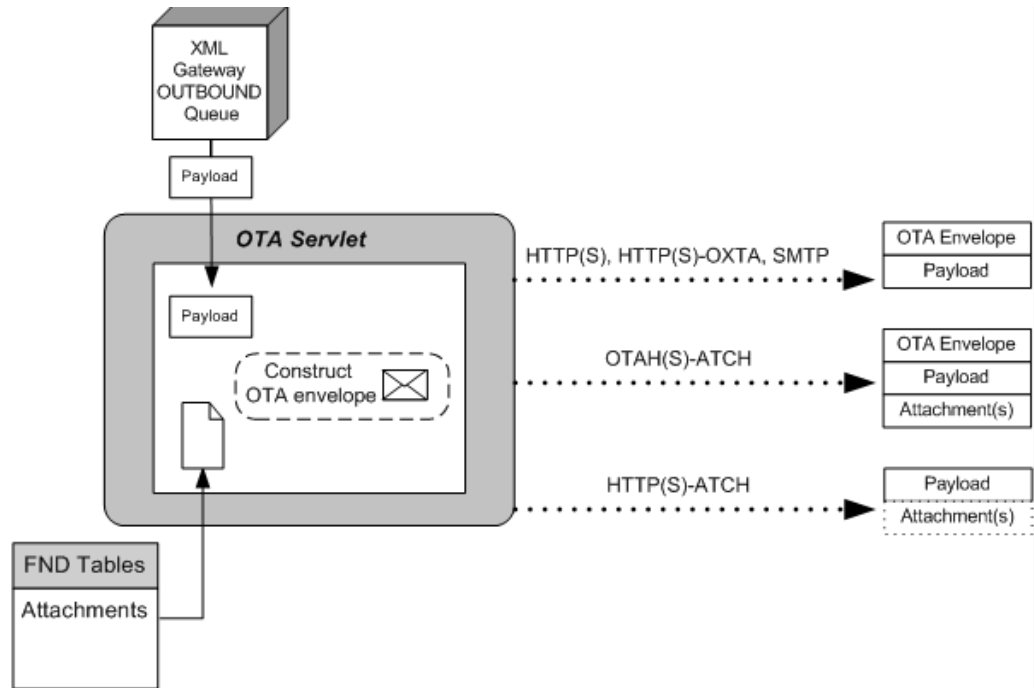
Each message enabled for a Trading Partner includes a protocol type (also known as communication method) that identifies the message delivery method. The associated correlation ID is determined internally and is associated with the message enqueued onto the agent (queue). Listeners defined for the agent will dequeue messages based on the correlation ID defined for it. For example, Oracle Transport Agent will only dequeue messages with a correlation ID of OXTA.

The following table shows the Protocol Type, Correlation ID, and Message System.

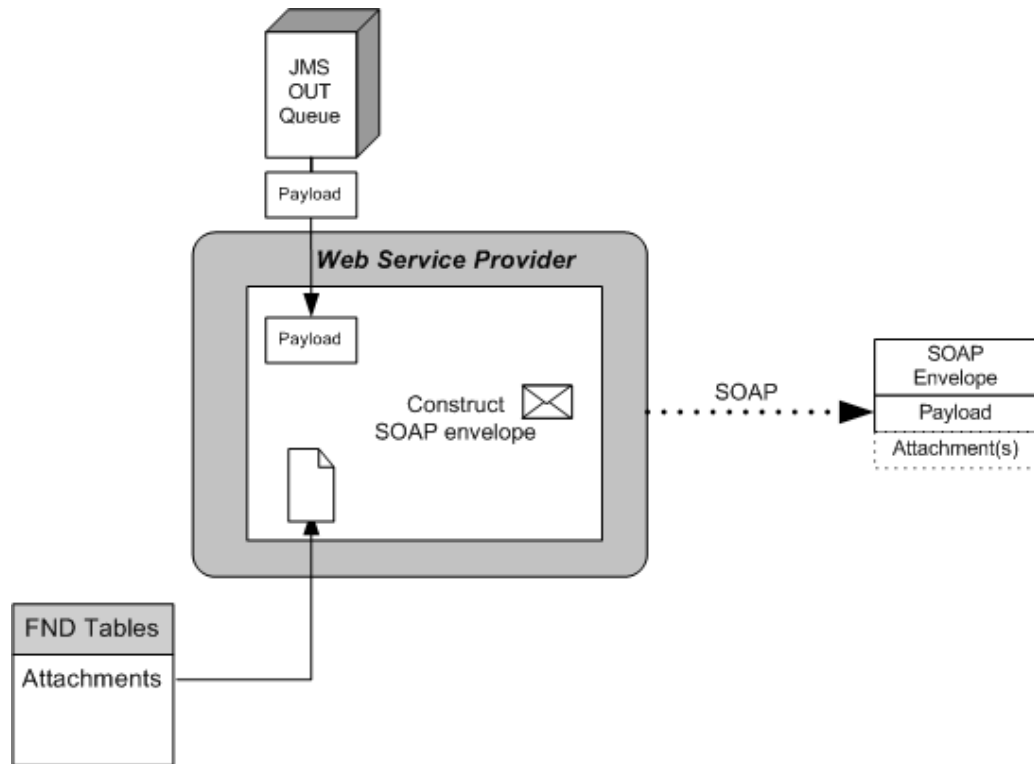
Protocol Type	Correlation ID	Message System
NONE	N/A	Message disabled
HTTP, HTTP-OXTA, HTTPS, HTTPS-OXTA, SMTP	OXTA	Oracle Transport Agent without attachments
OTAH-ATCH, OTAHS-ATCH	OXTA	Oracle Transport Agent with attachment using standard OTA envelope
HTTP-ATCH, HTTPS-ATCH	OXTA	Oracle Transport Agent with attachment and no OTA envelope
ITG03	ITG03	Oracle iProcurement Connector, must be enabled
IAS	IAS	Oracle Application Server, must be enabled
HTTP-WM, HTTPS-WM	WebMethods	Third party system available only to users of Oracle Exchange
SOAP	N/A	Web Service Agent
JMS	N/A	JMS Provider

The following graphics display the construction for each protocol type:

- For protocol types HTTP(S) and HTTP(S)-OXTA, the OTA servlet dequeues the message payload and constructs the OTA envelope.
- For protocol type OTAH(S)-ATCH, the OTA servlet dequeues the message payload, constructs the OTA envelope, and assembles the attachment(s).
- For protocol type HTTP(S)-ATCH, the OTA servlet dequeues the message payload and assembles any attachments, if present.



For protocol type SOAP, the Web Services provider dequeues the message payload, assembles any attachments, and constructs the SOAP envelope.



## XML Gateway Envelope

In addition to the business document such as a purchase order or invoice in the XML Payload, a set of message attributes are transmitted. Collectively, these attributes are called the XML Gateway envelope.

This section discusses the XML Gateway envelope and its data in the validation process for inbound messages, or its source of data for its creation for outbound messages. Data entered into the Trading Partner Setup form is referred to as data in the trading partner table. Data entered into the Define Transactions form is referred to as data in the transaction table.

Most of the data elements are copied from the Trading Partner tables or the Transaction tables to the XML Gateway envelope.

Transaction direction is determined by the XML Gateway. The direction values used by the XML Gateway are the following:

- IN for inbound messages
- OUT for outbound messages

The XML Gateway envelope consists of the data presented in the following table:

Attribute	Contents	Sample Values	Data Source
MESSAGE_TYPE	Payload message format	XML	(Hard-coded)
MESSAGE_STANDARD	Message format standard	OAG	Transaction table
TRANSACTION_TYPE	External Transaction Type for that business document	INVOICE	Trading Partner table
TRANSACTION_SUBTYPE	External Transaction Subtype for that business document	PROCESS	Trading Partner table
DOCUMENT_NUMBER	Business document number such as invoice number	(Not Used)	N/A
PARTYID	(Not Used)	(Not Used)	Trading Partner table. This is not used by the XML Gateway.
SOURCE_TP_LOCATION_CODE	Source Trading Partner Location Code	ACME_CHICAGO	SOURCE TP LOCATION (if not recreated) or TARGET TP LOCATION CODE (if recreated) from Trading Partner table
PARTY_TYPE	(Not Used)	(Not Used)	(Not Used)
PROTOCOL_TYPE	Transmission Protocol	SMTP, HTTP, HTTP-WM	Trading Partner table
PROTOCOL_ADDRESS	Transmission Address	me@co.com, http://www.co.com:5555	Trading Partner table
USERNAME	User Name	User1	Trading Partner table if Connection is "DIRECT"; Hub Definition if "HUB"
PASSWORD	password	*****	Trading Partner table if Connection is "DIRECT"; Hub Definition if "HUB"
ATTRIBUTE1	(determined by application)	N/A	(determined by application)
ATTRIBUTE2	(determined by application)	N/A	(determined by application)
ATTRIBUTE3	Rerouting Location (used only if the message is rerouted)		TARGET TP LOCATION CODE from Trading Partner table
ATTRIBUTE4	(determined by application)	N/A	(determined by application)
ATTRIBUTE5	(determined by application)	N/A	(determined by application)
PAYLOAD	XML business document	(XML Message)	

## MESSAGE\_TYPE

Payload message format. This defaults to "XML".

## **MESSAGE\_STANDARD**

(Defaults to OAG.) Message format standard as displayed in the Define Transactions form and entered in the Define XML Standards form.

## **TRANSACTION\_TYPE**

External Transaction Type for the business document from the Trading Partner table.

## **TRANSACTION\_SUBTYPE**

External Transaction Subtype for the business document from the Trading Partner table.

## **DOCUMENT\_NUMBER**

The document identifier used to identify the transaction, such as a purchase order or invoice number. This field is not used by the XML Gateway, but it may be passed on inbound messages.

## **SOURCE\_TP\_LOCATION\_CODE**

Source Trading Partner Location Code if no data is found in the Destination Trading Partner Location Code in the Trading Partner table.

## **PROTOCOL\_TYPE**

Transmission Protocol as defined in the Trading Partner table.

## **PROTOCOL\_ADDRESS**

Transmission address as defined in the Trading Partner table.

## **USERNAME**

USERNAME as defined in the Trading Partner table.

## **PASSWORD**

The password associated with the USERNAME defined in the Trading Partner table.

## **ATTRIBUTE3**

For outbound messages, this field has the value from the Destination Trading Partner Location Code in the Trading Partner table.

For inbound messages, the presence of this value generates another XML message that is sent to the trading partner identified in the Destination Trading Partner Location Code in the Trading Partner table. This value must be recognized by the hub to forward the XML message to the final recipient of the XML Message. Refer to XML Gateway Setup, page 3-1 for details.

## **PAYLOAD**

The XML message.



## Parameters defined by the Application

The following parameters may be defined by the base application:

- ATTRIBUTE1
- ATTRIBUTE2
- ATTRIBUTE4
- ATTRIBUTE5

## Parameters Not Used

The following parameters are not used:

- PARTYID
- PARTYTYPE

## Trading Partner Validation for Inbound Messages

### Events Raised in Oracle Workflow Business Event System

Oracle Workflow Business Event System is a tool used to integrate with the Oracle e-Business Suite for event-based XML message creation and consumption.

#### **Before XML Gateway processes the message:**

Messages sent by the trading partner are staged onto a message queue. An Oracle Workflow listener dequeues the message from the message queue and raises an event to the XML Gateway to begin processing.

Refer to Message Queues, page 5-1 for details.

#### **After XML Gateway processes the message:**

When XML Gateway completes processing the data according to the instructions in the message map associated with the trading partner, an event is published by XML Gateway to inform the Oracle e-Business Suite that it has successfully processed the inbound message. Any Oracle e-Business Suite module interested in this event may register a subscription to continue with the transaction.

## XML Gateway Processing

Inbound messages can take two paths:

- Standard XML Messages

If ATTRIBUTE3 in the XML Gateway envelope is NULL, the XML message is processed according to the message map.

- "Pass-Through" Messages

If ATTRIBUTE3 in the XML Gateway envelope has data, the inbound message will be recreated as an outbound message according to a message map for the trading partner identified in ATTRIBUTE3. This transaction is referred to as a "pass-through" message and defined as a message with "Dynamic Routing." Refer to Static and Dynamic Routing, page 3-20 for details.

## Standard XML Messages

The following table summarizes the required data in the XML Gateway envelope to identify standard XML messages. How this data is validated for an inbound message is described below.

Attribute	Sample Values	Used in Trading Partner Lookup
MESSAGE_STANDARD	OAG	YES
TRANSACTION_TYPE	INVOICE	YES
TRANSACTION_SUBTYPE	PROCESS	YES
PARTY_SITE_ID	ACME_CHICAGO	YES
ATTRIBUTE3	NULL	NO
PAYLOAD	(XML MESSAGE)	N/A

**Note:** USERNAME and PASSWORD can be placed in the XML Gateway envelope by the system that created the XML message. The transport agent uses the USERNAME and PASSWORD for delivery purposes. These fields are not passed to the XML Gateway from the transport agent software.

## "Pass-Through" XML Messages

XML Gateway can recreate, then route messages to another trading partner without loading the transaction into a base Oracle Application. This case is recognized by the presence of a trading partner code in ATTRIBUTE3 in the XML Gateway envelope in the inbound message.

If a trading partner location code is identified in ATTRIBUTE3 for an inbound message the following happens:

- A Trading Partner Setup table entry for an *outbound message* for this same TRANSACTION\_TYPE and TRANSACTION\_SUBTYPE is found for the Trading Partner identified in ATTRIBUTE3. The values in ATTRIBUTE3 must match an entry in the Destination Trading Partner Location Code defined in the Trading Partner Setup form.
- If the entity in ATTRIBUTE3 is found in the Trading Partner Setup tables, then another XML Gateway message is created according to the message map for that trading partner. This new XML message is then routed to the trading partner identified in the original ATTRIBUTE3.

The following table summarizes the required data in the XML Gateway envelope to identify these messages, and key data for the pass through XML message.

Attribute (Original XML Message)	Sample Inbound Message Values	Pass-Through Message Created (Find its corresponding Trading Partner Setup for an OUTBOUND message where the Trading Partner is identified in ATTRIBUTE3 of the original message)
MESSAGE_STANDARD	OAG	OAG
TRANSACTION_TYPE	INVOICE	INVOICE
TRANSACTION_SUBTYPE	PROCESS	PROCESS
PARTY_SITE_ID	ACME_CHICAGO	<b>BETA-LONDON</b> (from original ATTRIBUTE3)
ATTRIBUTE3 (the DESTINATION TRADING PARTNER LOCATION CODE in the Trading Partner Setup table. This becomes the PARTY_SITE_ID for the outbound message.)	BETA-LONDON	
PAYLOAD	(XML Message)	(XML Message)

## Validation Against Data in the Trading Partner Setup Form

Inbound messages validate data only against data that was entered in the Trading Partner Setup form.

### Source Trading Partner Location Code

Source Trading Partner Location Code is the code found in the XML envelope in PARTY\_SITE\_ID to identify the source of the message. This field must contain a value stored in Source Trading Partner Location Code in the Trading Partner setup table.

When a trading partner is entered into the Trading Partner Setup form, the Trading Partner Name and Trading Partner Site are selected from a list of values obtained from Oracle Receivables, Oracle Payables, or Oracle HR Locations. Associated with each Trading Partner site is a table ID in the base Oracle Application. See Derive Address ID from Location Code action, page 2-63 and Get Predefined Variable Value action, page 2-76 if you wish to use this table ID in a message map.

The parameters listed in the table below compose a search key against the Trading Partner table. This key is used to determine if the Trading Partner is enabled for that transaction, then to retrieve the message map name to process the transaction.

Search Parameters	Description	Sample
PARTY_SITE_ID	Source Trading Partner Location Code	ACME_CHICAGO
TRANSACTION_TYPE	External Transaction Type for that business document	INVOICE
TRANSACTION_SUBTYPE	External Transaction Subtype for that business document	PROCESS
DIRECTION	Determined by the XML Gateway	IN

The following table shows data that is returned from a search against data that was entered in the Trading Partner Setup form:

Returned Data	Description	Sample
Trading Partner Site enabled for the message	This data is stored in the table given the trading partner that was selected. This data is not displayed.	12345 (a table ID)
Message Map to use for this inbound transaction for the given trading partner		OAG_IN_INVOICE

Note that transaction direction is determined by the XML Gateway. The direction values used by the XML Gateway are the following:

- IN for inbound messages
- OUT for outbound messages

## Trading Partner Validation for Outbound Messages

### Event Raised in Oracle Workflow Business Event System

Application business events for outbound messages are raised by the Oracle e-Business Suite module responsible for the transaction. The raise occurs at an application point of interest, such as when a purchase order is approved or an invoice is confirmed.

### Event Subscription Defined in Oracle Workflow Business Event System

A corresponding event subscription is defined to consume the event. The event subscription may be based on the XML Gateway Rule Function or the Workflow Default Rule Function. Embedded in the XML Gateway Rule Function is a check to determine if the Trading Partner is defined and the message is enabled.

With the Workflow Default Rule Function, the Transaction Delivery Required function activity is used to make the same evaluation. The outbound document is generated and sent only if the Trading Partner is defined and the transaction is enabled. Which approach is used depends on what is known about the Trading Partner before the subscription is executed.

See Integrating Oracle XML Gateway with Oracle Workflow Business Event System, page 6-1.

## How to Implement the OAG Confirmation Business Object Document

### Purpose of the Confirmation Message

The purpose of the confirmation message is to communicate the status of a business document. In addition to providing general document status information, details regarding any errors detected may be included in the confirmation message.

The confirmation message is a general-purpose document that may be used to acknowledge any business document. It may be used in addition to but not as a substitute for specific purpose acknowledgements such as Purchase Order Acknowledgment or Purchase Order Change Acknowledgment documents.

Refer to [www.openapplications.org](http://www.openapplications.org) for details regarding the Confirm Business Object Document (BoD) DTD.

### Structure of the Confirmation Message

The confirmation message contains the following two parts:

- CONFIRM\_BOD/CONFIRM

The CONFIRM data type is used to describe the business document it is acknowledging. Included are the document status, description of the document, and an identifier for the original business document.

- CONFIRM\_BOD/CONFIRMMSG

The CONFIRMMSG data type is used to provide a detailed description of the document status. Included are the description and any reason codes. Each CONFIRM data type may have many CONFIRMMSG data types associated with it.

### XML Gateway Seeded Confirmation Message Maps

Oracle XML Gateway provides seeded maps for both the inbound and outbound Confirm BoD message. The names of the seeded message maps are as follows:

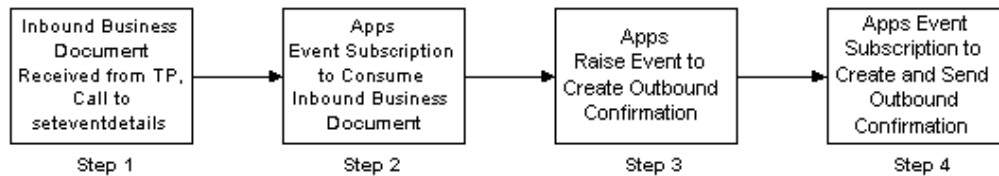
- ECX\_CBODI\_OAG72\_IN\_CONFIRM
- ECX\_CBODO\_OAG72\_OUT\_CONFIRM

The XML Gateway seeded maps are incorporated into the Oracle E-Business Suite application business process where necessary.

### E-Business Suite Seeded Events and Event Subscriptions

Usage of the Confirm BoD varies between application modules of the Oracle E-Business Suite. For application modules that have incorporated the Confirm BoD into their business processes, the process is modeled as follows:

## Outbound Confirmation



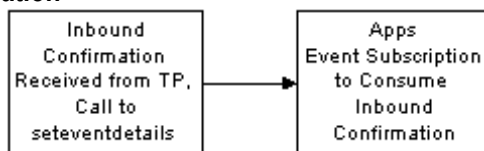
Oracle XML Gateway sends an outbound confirmation message in response to an inbound business document.

The inbound business document initiates the process (Step 1). As with all inbound messages, XML Gateway notifies the Oracle E-Business Suite application that it has processed an inbound business document.

The application event subscription defined for the inbound document is executed (Step 2) and is followed by a raise event (Step 3) to notify XML Gateway to create and send the outbound confirmation (Step 4).

Steps 2 and 3 are defined separately to allow for application business processes between the consumption of the inbound business document and the creation and delivery of the outbound confirmation message.

## Inbound Confirmation



Oracle XML Gateway will process an inbound confirmation message received from a the Trading Partner in response to an outbound business document. It will notify the Oracle E-Business Suite that it has processed an inbound confirmation message.

The application event subscription defined for the inbound confirmation message is executed. The behavior of the event subscription will vary by application module and is dependent on whether the application module is interested in negative, positive, or both types of confirmations.

## How to Implement or Disable a Seeded Confirmation Message

The application event subscriptions to create and send an outbound message or to consume an inbound message are delivered by the Oracle E-Business Suite application modules.

The seeded event name for the inbound or outbound Confirm BoD message is as follows:

ORACLE.APPS.<COMPONENT>.<TASK>.CONFIRM

COMPONENT is the internal transaction type entered on the Define Transactions form. It represents the product short name.

TASK is the internal transaction subtype entered on the Define Transactions form. It represents a description of the object.

An example of an event name for a confirmation event associated with an outbound purchase order is: ORACLE.APPS.PO.POO.CONFIRM.

The corresponding seeded event subscription is defined as enabled. Use the Oracle Workflow Administrator: Add Events/Event Group window to view or disable the seeded events and event subscriptions if you do not wish to implement the Confirm BoD message.

To implement the seeded event subscriptions, define the Trading Partner and enable the confirmation message as follows:

1. Define the Trading Partner using the XML Gateway Define Trading Partners form.
2. Enable the business document and set the Document Confirmation flag to one of the following:

1 = Send confirmation only if there are errors

2 = Always send confirmation

**Important:** Ensure that for an outbound business document you enable an inbound confirmation document (Transaction Subtype CBODI); and for an inbound business document you enable an outbound confirmation document (Transaction Subtype CBODO).

**Note:** Make sure that the value for Source Trading Partner Location Code is the same for the original business document and its corresponding confirmation document.

Refer to Trading Partner Setup, page 3-13 for more details on setting up the Trading Partner to implement the confirmation message.





---

## Message Queues

This chapter introduces the message queues.

This chapter covers the following topics:

- Queues

### Queues

Queues are tables on a database that are managed by Oracle Advanced Queuing.

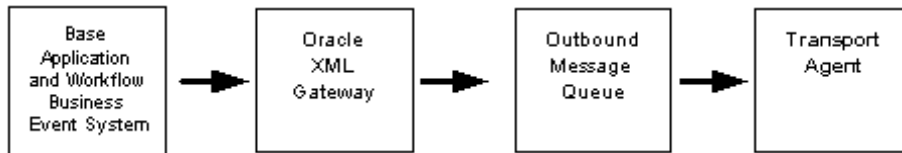
The XML Gateway uses queues specifically at two points in the process, as well as employing a general error queue. The first point is at the transport agent level between the transport agent module and the XML Gateway. The second point is at the transaction level between base Oracle E-Business Suite products or other source process, and the XML Gateway. Details about these queues are provided in the table below.

Queue Name	Description	Level and Position in Process
ECX_INBOUND	<b>Inbound Message Queue:</b> Holds all messages that enter the process through the Transport Agent, or are placed directly on the queue by an API.	Transport Agent Level Between the Transport Agent and the Inbound Transaction Queue
ECX_OUTBOUND	<b>Outbound Message Queue:</b> XML Gateway enqueues all outbound messages that it formatted on this queue.	Transport Agent Level Between the XML Gateway and the Transport Agent
ECX_IN_OAG_Q	<b>Inbound Transaction Queue:</b> Holds inbound messages that originated from the ECX_INBOUND queue, then enqueued on this queue by Oracle Workflow. These messages will be processed by the XML Gateway.	Intermediate Level After the message is received through the Transport Agent (Transport Agent Level above), Oracle Workflow enqueues the message in this queue for the XML Gateway to process.
WF_ERROR	<b>Workflow Error Queue:</b> For errors detected by XML Gateway or WF BES. Notifications are sent to a Trading Partner contact or the System Administrator.	Transaction Level

Refer to the Oracle 9i Database title *Application Developer's Guide - Advanced Queueing* for details, including topics on checking the status and content of the queues.

## Outbound Queues

### Outbound Message Queues



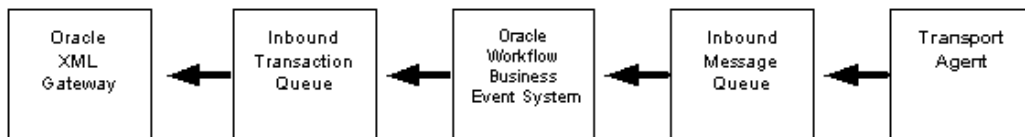
The outbound Message Queue is positioned between the XML Gateway and the Transport Agent.

The XML Gateway creates XML messages, then enqueues them on this queue.

Next Step: The Transport Agent dequeues the message and delivers it to the Trading Partner.

## Inbound Queues

### Inbound Message Queue



Inbound messages are first queued on the Inbound Message Queue.

The inbound Message Queue is positioned between the Transport Agent and the Oracle Workflow Business Event System.

There are two methods for placing a message on the Inbound Message Queue:

- The Transport Agent enqueues the message.
- An API writes directly to the queue.

Refer to the Oracle 9i Database title *Application Developer's Guide - Advanced Queueing* for details.

The full message must be formatted according to the XML Gateway envelope message format described in the Execution Engine section. See XML Gateway Envelope , page 4-6 for details.

Next Step: Oracle Workflow Business Event System will copy the inbound messages to the proper inbound Transaction Queue. One queue is seeded, but you may define other queues to meet your business needs. See information below.

### Inbound Transaction Queue

This is the second queue for an inbound message.

After the message is received through the inbound Message Queue, Oracle Workflow Business Event System enqueues the message to this queue for the XML Gateway to process.

Next Step: The XML Gateway will process the inbound messages placed in this queue.

## Error Queue

When an error is detected by Oracle XML Gateway or Oracle Workflow Business Event System the message is enqueued on to the error queue. A Workflow listener dequeues the error and sends a notification to the Trading Partner contact for data errors or to the System Administrator contact for system or process errors.

## Oracle Transport Agent Send Inbound HTML Page

To verify that a message can be placed on a queue after setup, use the Oracle Transport Agent Send Inbound page. This page allows you to send an Inbound HTML document to the Oracle Transport Agent running on the same Web server as the HTML page.

The ECXOTAINbound.html file included with the Oracle Transport Agent allows you to send XML documents inbound from a Web server. The document will be received by OTA and placed on the ECX\_INBOUND queue. This HTML file is included for testing purposes and should not be used in a production environment.

To use the ECXOTAINbound.htm page, open a Web browser and enter the following URL:

```
http://<server name>:<port>/OA_HTML/US/ECXOTAINbound.htm
```

The page will prompt you for the input parameters used by the OTA messaging protocol. Refer to the *Oracle MetaLink* document "Understanding the OTA (Transport Agent) Protocol" for information regarding the OTA messaging protocol parameters.

Column Name	Description
TRANSPORT_PROTOCOL	Defaults to OXTA.
TRANSPORT_PROTOCOL_VERSION	Defaults to 1.0.
REQUEST_TYPE	Select "Send".
MESSAGE_ID	Enter the unique reference number that identifies the transaction.
MESSAGE_TYPE	Defaults to XML.
MESSAGE_STANDARD	Defaults to OAG.
TRANSACTION_TYPE	(Required) Enter the type of transaction, such as PO for Purchase Order Inbound.
TRANSACTION_SUBTYPE	(Required) Enter the subtype of the transaction.
DOCUMENT_NUMBER	Enter the document number.
PARTYID	(Optional) Enter the trading partner ID for the sender.
SOURCE_TP_LOCATION_CODE	(required) Enter the Source Trading Partner Location Code, page 3-18 (should match the Trading Partner Details column of the same name).
PROTOCOL_TYPE	Not required for inbound messages.
PROTOCOL_ADDRESS	Required only if REQUEST_TYPE is EME.
USERNAME	(Required) Enter a valid user name for the receiving system.
PASSWORD	(Required) Enter the password for the user name entered.
ATTRIBUTE1	Optional
ATTRIBUTE2	Optional
ATTRIBUTE3	Required for pass-through transactions only.
ATTRIBUTE4	Optional
ATTRIBUTE5	Optional
PAYLOAD	Enter the message payload.

## XML Gateway Message Format

XML messages that are enqueued or dequeued for processing by XML Gateway must conform to the system.ecxmsg data type.

The system.ecxmsg data type consists of the following:

<b>Name</b>	<b>Type</b>
MESSAGE_TYPE	VARCHAR2(80)
MESSAGE_STANDARD	VARCHAR2(80)
TRANSACTION_TYPE	VARCHAR2(80)
TRANSACTION_SUBTYPE	VARCHAR2(80)
DOCUMENT_NUMBER	VARCHAR2(80)
PARTYID	VARCHAR2(80)
SOURCE_TP_LOCATION_CODE	VARCHAR2(80)
PARTY_TYPE	VARCHAR2(80)
PROTOCOL_TYPE	VARCHAR2(200)
PROTOCOL_ADDRESS	VARCHAR2(200)
USERNAME	VARCHAR2(200)
PASSWORD	VARCHAR2(200)
ATTRIBUTE1	VARCHAR2(200)
ATTRIBUTE2	VARCHAR2(200)
ATTRIBUTE3	VARCHAR2(200)
ATTRIBUTE4	VARCHAR2(200)
ATTRIBUTE5	VARCHAR2(200)
PAYLOAD	CLOB

Refer to XML Gateway Envelope, page 4-6 for a description of each field.

Non-Oracle messaging systems can enqueue an XML message for the Oracle E-Business Suite to process. This is done by defining an object of the same data type, populating the object, then calling an Advanced Queuing enqueue API to enqueue the object. The Oracle E-Business Suite Transport Agent will receive the object and deposit it onto the ECX\_OUTBOUND agent. From there, the standard inbound agent listener assumes control of the process.



---

# Integrating Oracle XML Gateway with Oracle Workflow Business Event System

This chapter covers the following topics:

- Integrating Oracle XML Gateway with Oracle Workflow Business Event System
- Oracle Workflow Builder - Item Types
- XML Gateway Standard Item Type
- XML Gateway Error Processing Item Type
- Configure Oracle Prebuilt Inbound Messages
- Configure Oracle Prebuilt Outbound Messages
- Application to Application Integration
- Manage Workflow Processes
- Monitor Workflow Processes
- Development Guidelines for Custom Messages for B2B Integration
- Common Questions
- Message Delivery Status

## Integrating Oracle XML Gateway with Oracle Workflow Business Event System

### Overview

Oracle Workflow is a process management tool used to seamlessly integrate Oracle E-Business Suite business processes together. Introduced with Oracle Workflow 2.6 is the Business Event System to further support business process integration at the business event level.

Integral to the Business Event System is the ability to raise a business event to signal that something of interest has occurred in the Oracle E-Business Suite. Using Oracle Purchasing as an example, an event of interest would be the creation, change, confirmation, or deletion of a purchase order.

Complementary to the raise event is an event subscription defined to perform some activity when it detects an event of interest has occurred. An example of an event subscription in our Oracle Purchasing scenario is to send a message to the

appropriate supplier when a new order is created, or when an existing order is changed, confirmed, or deleted.

An event subscription is controlled by a rule function. The result of the rule function determines whether the associated Workflow process will be executed. The actual activity performed by the event subscription is based on the Workflow process defined for it.

In the context of the integration between the Oracle Workflow Business Event System and Oracle XML Gateway, business events and their corresponding event subscriptions are seeded and deployed to the Oracle E-Business Suite. Once the Trading Partner is defined, the seeded business events and event subscriptions may be implemented as is, or they may be configured to perform specific activities relevant to the business requirements.

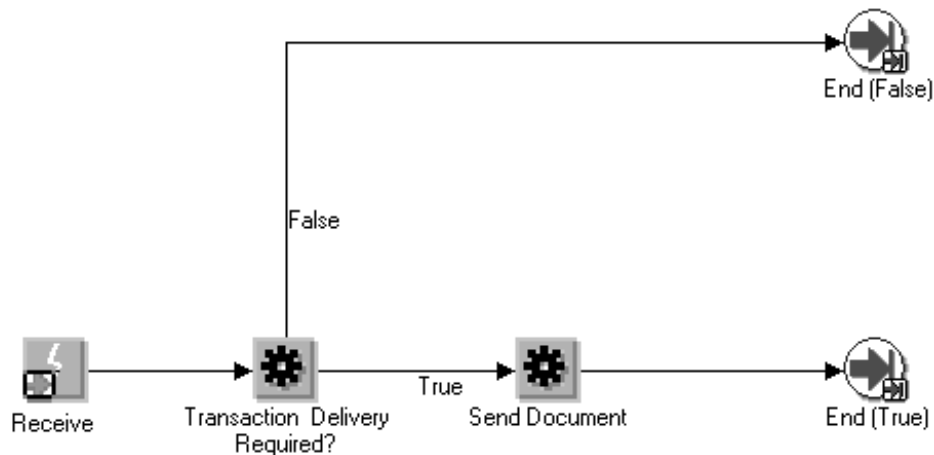
Application business events for outbound messages are raised by the Oracle E-Business Suite application module responsible for the message and processed by the event subscription defined for it. The seeded event subscription for the B2B integration may be based on the XML Gateway Rule Function or the Workflow Default Rule Function.

Embedded in the XML Gateway Rule Function is the Transaction Delivery Required Function used to determine if the Trading Partner is enabled for the transaction. The XML Gateway Rule Function can therefore be simply represented as follows:



The Receive Event is followed by the Send Document function activity which proceeds to End.

The Workflow Default Rule Function uses the Transaction Delivery Required function as a process activity to determine if the Trading Partner is enabled for the transaction. This process consists of the following steps: The Receive Event is followed by the Transaction Delivery Required function activity. If it returns TRUE, the Send Document function activity is executed. If FALSE, the process proceeds to End. This process is shown below:





With either rule function, no message is sent if the Trading Partner is not enabled for the transaction.

For efficiency, the XML Gateway Rule Function is the recommended approach as it has already predetermined that the Trading Partner is enabled for the transaction before it proceeds with generating the message. However, the Workflow Default Rule function approach provides a visual audit trail of the Workflow process.

With either approach, Oracle XML Gateway gathers the data from the Oracle E-Business Suite, creates the XML message, and delivers it to the Trading Partner for Business to Business (B2B) integration or to another application module for Application to Application (A2A) integration.

Inbound messages received from the Trading Partner for Business to Business (B2B) integration are enqueued onto the inbound queue. Inbound messages received from another application module for Application to Application (A2A) integration are stored in the Event Message attribute (of the Consume XML Document function) to be accessed by the XML Gateway execution engine for processing.

An XML Gateway inbound rule function is executed in the B2B integration to validate the Trading Partner and to identify the transaction queue. If the Trading Partner is valid, the message is enqueued onto the transaction queue for processing by the XML Gateway execution engine.

The XML Gateway execution engine processes the data according to the instructions in the message map associated with the Trading Partner and message. If successful, the message map sets the Event Details to inform the Oracle E-Business Suite of the Event. It is the application of the map to the message that enables you to determine what the Event Details are. The event is then processed by an application event subscription interested in the inbound message.

Oracle Workflow Business Event System consists of three key components that are used to integrate with the Oracle E-Business Suite to create and consume XML messages:

- The Oracle Workflow Builder contains the Oracle Workflow Standard Item Type and the Oracle XML Gateway Standard Item Type. The components of the standard item types can be used to define Workflow processes.
- Oracle Workflow Administrator used to register new events and event subscriptions as well as configure seeded event subscriptions.
- The Oracle Workflow processing engine used to execute the workflow processes.

## Oracle Workflow Builder - Item Types

Delivered with XML Gateway are two standard item types:

- XML Gateway Standard Item Type
- XML Gateway Error Processing Item Type

An Item Type is a group of components of a particular category that share the same set of item attributes (also known as variables).

Both the XML Gateway Standard Item Type and XML Gateway Error Processing Item Type can be accessed using the Oracle Workflow Builder.

## Components of an Item Type

Common to all item types are the following components:

- Attributes
- Process Activity
- Notification Activity
- Function Activity
- Event Activity
- Messages
- Lookup Types

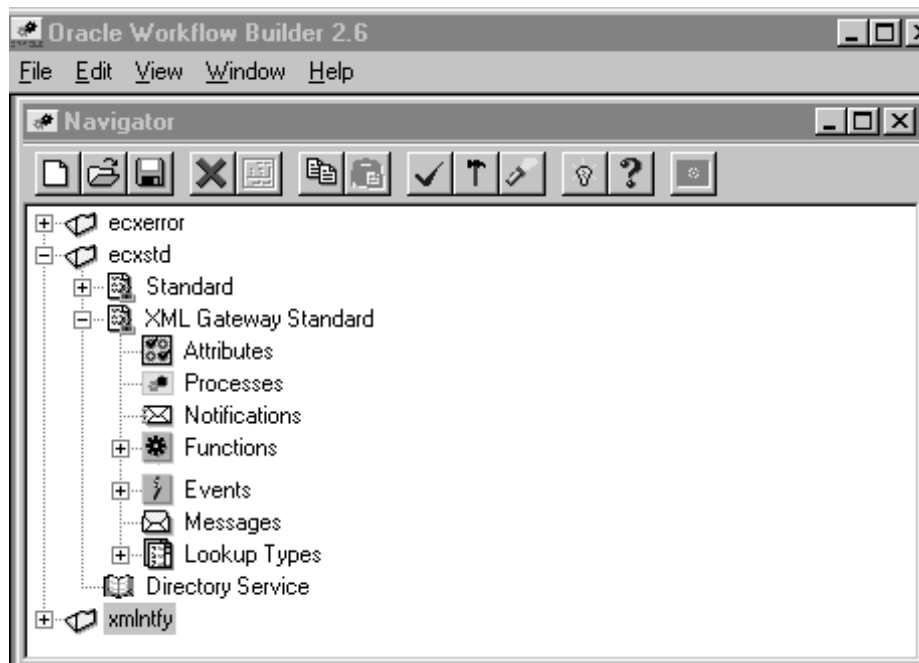
A Workflow process may be made up of event activities, function activities, notification activities, or process activities. A message is the information sent in a notification activity. Lookup types represent a list of valid values. Attributes represent variables used to share data between activities in the Workflow process.

See the *Oracle Workflow User's Guide* for more information.

## XML Gateway Standard Item Type

The XML Gateway Standard Item Type is a toolkit consisting of attributes, processes, functions, and event activities. Configure the seeded events and event subscriptions delivered by the Oracle E-Business Suite for prebuilt XML messages in support of B2B integration requirements.

In addition, the function activities can be used to define Workflow processes in support of A2A integration requirements.

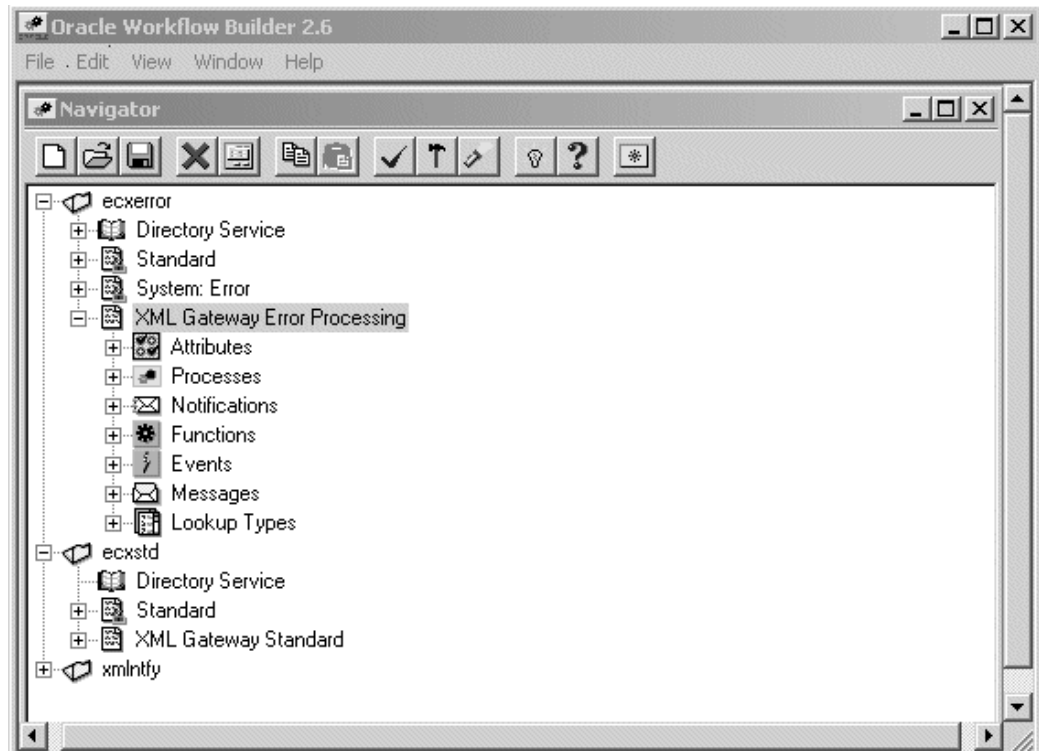


See XML Gateway Standard Item Type, page 6-5 for details regarding the function and event activities.

## XML Gateway Error Processing Item Type

The XML Gateway Error Processing Item Type contains error handling processes to manage errors detected by Oracle Workflow Business Event System or Oracle XML Gateway.

Oracle Workflow sends a notification to the Trading Partner contact for data errors or to the XML Gateway system administrator for system or process errors. For errors that require collaboration between the Trading Partner contact and the XML Gateway system administrator, a notification is sent to both parties to encourage discussion and to expedite problem resolution.



Refer to XML Gateway Error Processing Item Type, page 6-24 for details regarding the error handling process.

## E-Business Suite Application Module-Specific Item Type

The seeded event subscription delivered for Oracle prebuilt messages is delivered in an Oracle E-Business Suite application module-specific item type. The seeded event subscription may be configured using the function activities defined in the XML Gateway Standard Item Type or customized to support the application-specific requirement.

See Configure Oracle Prebuilt Inbound Messages, page 6-44 or Configure Oracle Prebuilt Outbound Messages, page 6-45 for details on how to configure a seeded event subscription.

## XML Gateway Standard Item Type

The XML Gateway Standard Item Type is a toolkit consisting of attributes, processes, functions, and event activities. These are provided to implementers to build Workflow

processes to configure the seeded event subscriptions delivered by the Oracle E-Business Suite for prebuilt XML messages in support of B2B integration requirements.

In addition, the function activities can be used to define Workflow processes in support of A2A integration requirements.

The Notifications and Messages are contained in the Oracle E-Business Suite application module-specific item type and are defined using the XML Gateway Error Processing Item Type.

See XML Gateway Error Processing Item Type, page 6-24.

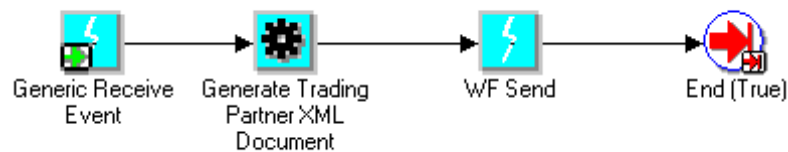
## Attributes

Attributes, also known as variables, are used to support the function, event, notification, and process activities defined for the XML Gateway Standard Item Type.

## Processes

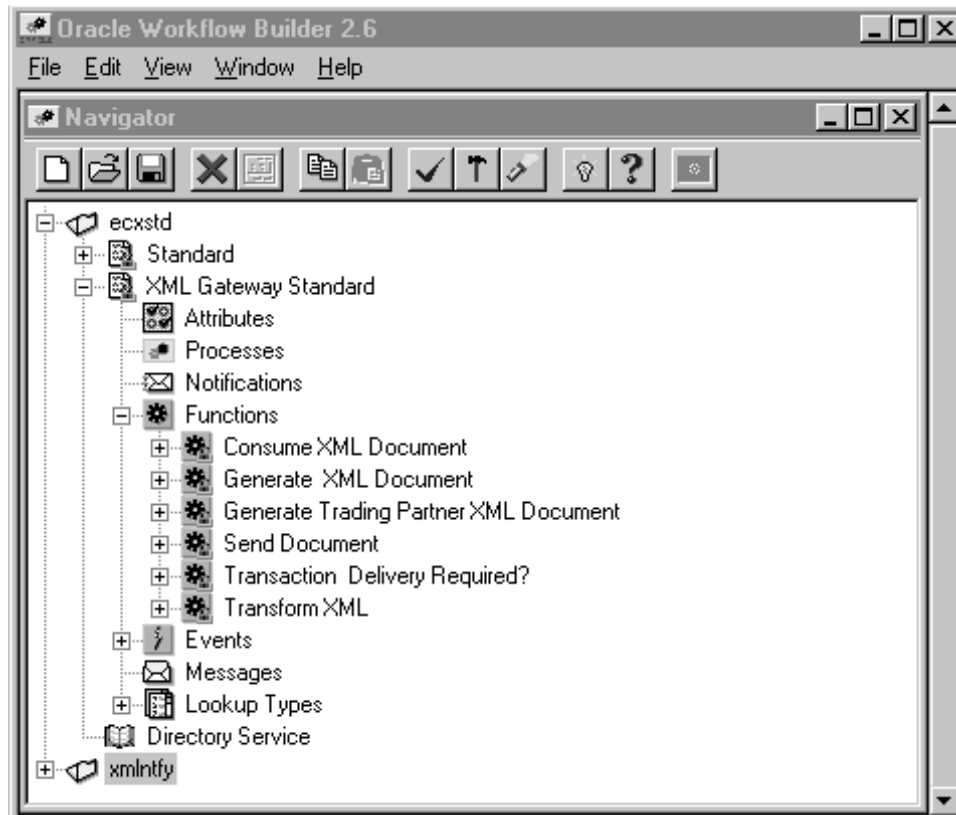
The XML Gateway Standard Item Type supports the following process:

- Create Trading Partner Message (for ECX internal use only)



The Create Trading Partner Message is a rule function available from the Workflow Event Subscription page.

## Functions



The XML Gateway Standard Item Type includes five function activities as shown in the following table:

Function Display Name	A2A/B2B	Inbound / Outbound	Purpose
Consume XML Document	A2A	Inbound	Trigger XML Gateway to process inbound message on the Event Message attribute.
Generate XML Document	A2A	Outbound	Create XML message and store in Event Message attribute. Trading Partner verification is not required.
Generate Trading Partner XML Document	B2B	Outbound	Verify transaction is enabled for the Trading Partner, create XML message, and store in Event Message attribute.
Send Document	B2B	Outbound	Create and enqueue XML message onto outbound queue.
Transaction Delivery Required	B2B	Outbound	Verify transaction is enabled for the Trading Partner.
Transform XML	A2A/B2B	Inbound/Outbound	Enable XML to XML transformations

Application-to-Application (A2A) exchange of data is conducted within and outside your enterprise to share or synchronize data between information systems. There is

no Trading Partner involved in an A2A integration. An example of an A2A exchange is to move data from the Oracle Payables module to Oracle General Ledger for GL consolidation.

Business-to-Business (B2B) exchange of data is conducted with a Trading Partner outside of your enterprise for which Trading Partner validation is required. Data will not be sent to or received from an invalid Trading Partner. An example of a B2B exchange is to send a purchase order to a supplier.

There is not a controlling Workflow process for inbound B2B exchanges. The XML Gateway execution engine subscribes directly to the inbound queue.

Functions will be referred to by their function display names. The internal function names are provided for reference in the following table:

Function Display Name	Internal Name
Consume XML Document	PROCESSXML
Generate XML Document	GETXML
Generate Trading Partner XML Document	GETTPXML
Send Document	DOCUMENT_SEND
Transaction Delivery Required	DELIVREQUIRED
Transform XML	XMLTOXML

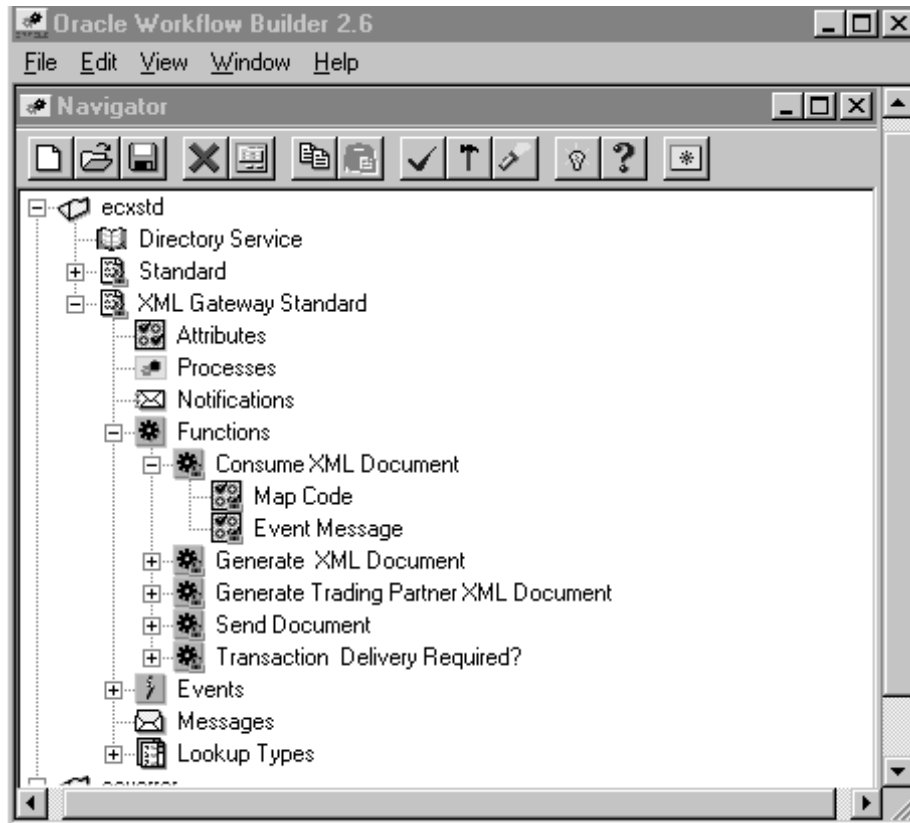
The following is a description of each function supported by the XML Gateway Standard Item Type.

### **Consume XML Document (applies to A2A for inbound messages)**

The Consume XML Document function triggers the XML Gateway Execution Engine to process the inbound message in the Event Message attribute. The XML Gateway execution engine will process the message according to the message map associated with the message, and will conclude by raising a business event to indicate that it has successfully processed the inbound message.

See Procedure Call: Execute Procedure, page 2-78 for details regarding the Procedure Call action to raise a business event.

The Consume XML Document function is for inbound messages in the A2A scenario only. The XML Gateway execution engine is automatically triggered in a B2B scenario when an inbound message arrives on the inbound queue.



The attributes for the Consume XML Document function are shown in the following table. The values for the attributes are provided by the business event, another function activity, or are set as constants.

Attribute Name	Attribute Description
Map Code	This is the Map Code defined in the Message Designer and associated to the Trading Partner that uniquely identifies the message map associated with the business document enabled for the Trading Partner.
Event Message	Event Message contains the event data as well as header properties for the event name, event key, and error data. The Event Message attribute is required to store the inbound message for processing by the XML Gateway execution engine.

### Generate XML Document (applies to A2A for outbound messages)

The Generate XML Document function is used to retrieve data from the Oracle E-Business Suite. The data for the XML message retrieved from the Oracle E-Business Suite is stored in the Event Message attribute. The Event Message is then processed according to the subsequent Workflow instruction, which may be to send it to another Oracle E-Business Suite application module or to the WF\_OUT agent.

The difference between the Generate XML Document function and the Generate Trading Partner XML Document function is that this function does not require Trading Partner validation since there is no Trading Partner involved in A2A integration.

The Generate XML Document function activity is provided for use in Workflow processes. For environments where a Workflow process does not exist, the ECX\_STANDARD.generate function can be called from the Workflow Define Event window, Generate Function field. The details of the function are as follows:

## **ECX\_STANDARD**

### **PL/SQL Syntax**

function generate

```
(p_event_name in varchar2,  
 p_event_key in varchar2  
 p_parameter_list in wf_parameter_list_t)  
return CLOB;
```

### **Description**

#### **Arguments (input)**

##### **p\_event\_name**

Unique identifier for the business event associated with the transaction.

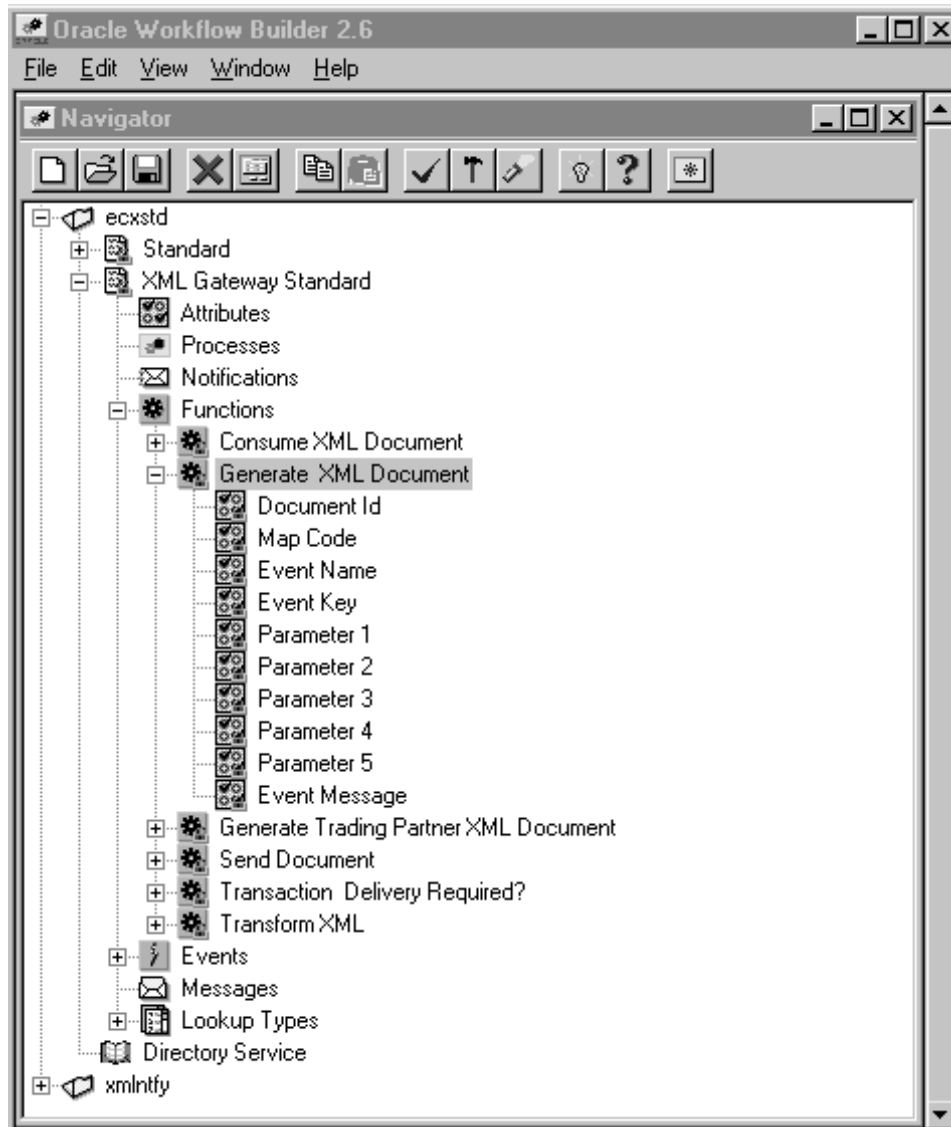
##### **p\_event\_key**

Unique identifier for the business document from the Oracle e-Business Suite associated with the business event.

##### **p\_parameter\_list**

Parameter list containing the document selection criteria.





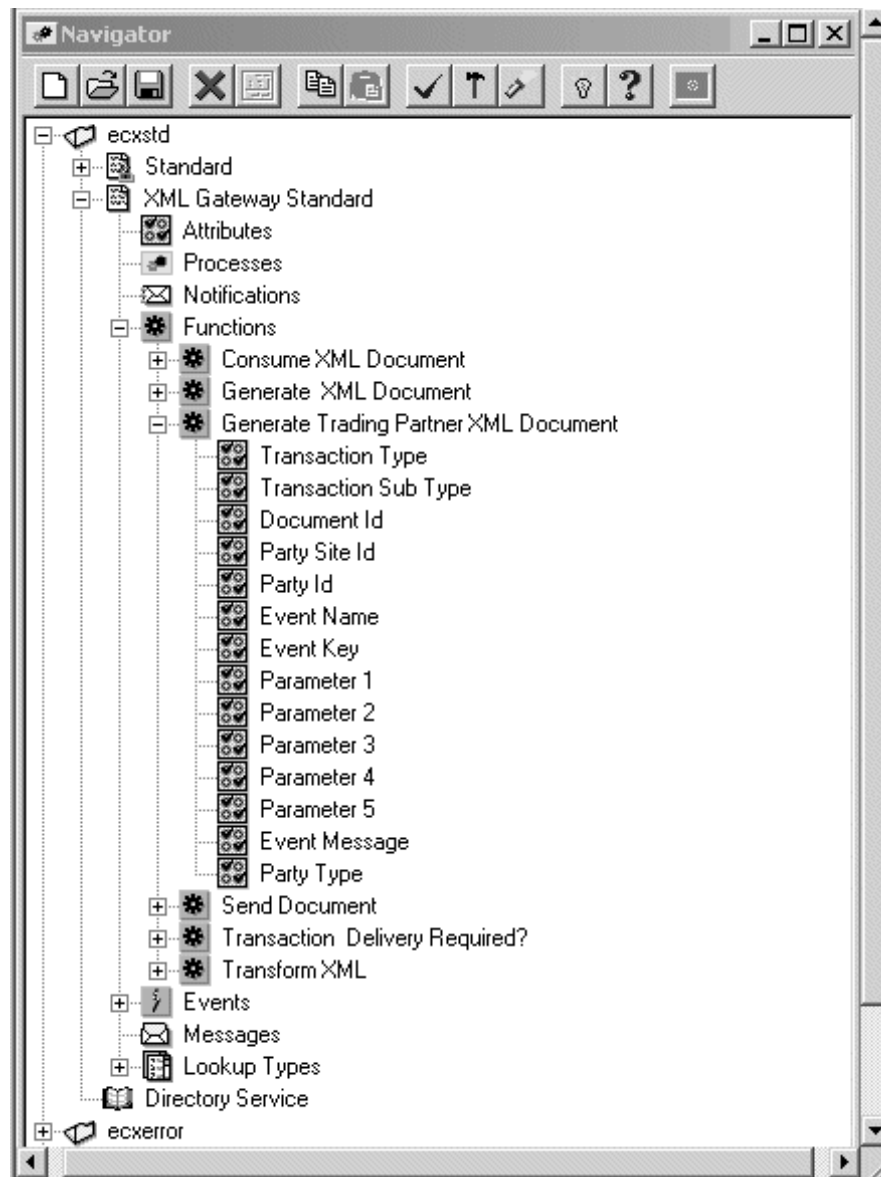
The attributes for the Generate XML Document function are shown in the following table. The values for the attributes are provided by the business event, another function activity, or are set as constants.

Attribute Name	Attribute Description
Document ID	This is the unique identifier for the business document. It can be the document number or its associated database key, whichever is guaranteed to be unique for the transaction. The Document ID is optional for the Generate XML Document function unless it is used in the message map as a selection criterion.
Map Code	This is the Map Code defined in the Message Designer and associated to the Trading Partner that uniquely identifies the message map associated with the business document enabled for the Trading Partner. The Map Code is required for the Generate XML Document function.
Event Name	This is a unique identifier for the business event. The naming convention is <code>oracle.apps.&lt;product code&gt;.&lt;component&gt;.&lt;object&gt;.&lt;event&gt;</code> . The Event Name is required for the Generate XML Document function to store the value returned.
Event Key	This is a unique identifier for an instance of an event. The combination of event name, event key, and event data fully describe what occurred in the event. The Event Key is required for the Generate XML Document function activity to store the value returned.
Parameter 1	Optional variable. This attribute is used if it is used in the message map.
Parameter 2	Optional variable. This attribute is used if it is used in the message map.
Parameter 3	Optional variable. This attribute is used if it is used in the message map.
Parameter 4	Optional variable. This attribute is used if it is used in the message map.
Parameter 5	Optional variable. This attribute is used if it is used in the message map.
Event Message	Event Message contains the event data as well as header properties for the event name, event key, and error data. The Event Message is required for the Generate XML Document function to store the value returned.

### Generate Trading Partner XML Document (applies to B2B for outbound messages)

The Generate Trading Partner XML Document function is used to retrieve data from the Oracle E-Business Suite. The data for the XML message retrieved from the Oracle E-Business Suite is stored in the Event Message attribute. The Event Message is then processed according to the subsequent Workflow instruction, which may be to send it to another Oracle E-Business Suite application module or to the WF\_OUT agent.

The difference between the Generate Trading Partner XML Document function and the Generate XML Document function is that this function is used in B2B integration for which a valid Trading Partner is required.



The attributes for the Generate Trading Partner XML Document function are shown in the following table. The values for the attributes are provided by the business event, another function activity, or are set as constants.

Attribute Name	Attribute Description
Transaction Type	This is the Transaction Type defined in the XML Gateway Define Transactions form. The Transaction Type is optional for the Generate Trading Partner XML Document function because it can be derived from the Map Code.
Transaction Subtype	This is the Transaction Subtype defined in the XML Gateway Define Transactions form. The Transaction Subtype is optional for the Generate Trading Partner XML Document function because it can be derived from the Map Code.
Document ID	This is the unique identifier for the business document. It can be the document number or its associated database key, whichever is guaranteed to be unique for the transaction. The Document ID is optional for the Generate XML Document function unless it is used in the message map as a selection criterion.
Party Site ID	This is the unique identifier for the Trading Partner site defined in Oracle E-Business Suite. The Party Site ID is required for the Generate Trading Partner XML Document function.
Party ID	This is the unique identifier for the Trading Partner defined in Oracle E-Business Suite. This field is optional.
Event Name	This is a unique identifier for the business event. The naming convention is <code>oracle.apps.&lt;product code&gt;.&lt;component&gt;.&lt;object&gt;.&lt;event&gt;</code> . The Event Name is required for the Generate Trading Partner XML Document function to store the value returned.
Event Key	This is a unique identifier for an instance of an event. The combination of event name, event key, and event data fully describe what occurred in the event. The Event Key is required for the Generate Trading Partner XML Document function to store the value returned.
Parameter 1	Optional variable. This attribute is used if it is used in the message map.
Parameter 2	Optional variable. This attribute is used if it is used in the message map.
Parameter 3	Optional variable. This attribute is used if it is used in the message map.
Parameter 4	Optional variable. This attribute is used if it is used in the message map.
Parameter 5	Optional variable. This attribute is used if it is used in the message map.
Event Message	Event Message contains the event data as well as header properties for the event name, event key, and error data. The Event Message is required for the Generate Trading Partner XML Document function to store the value returned.
Party Type	The Party Type defined in the XML Gateway Trading Partner Setup window.

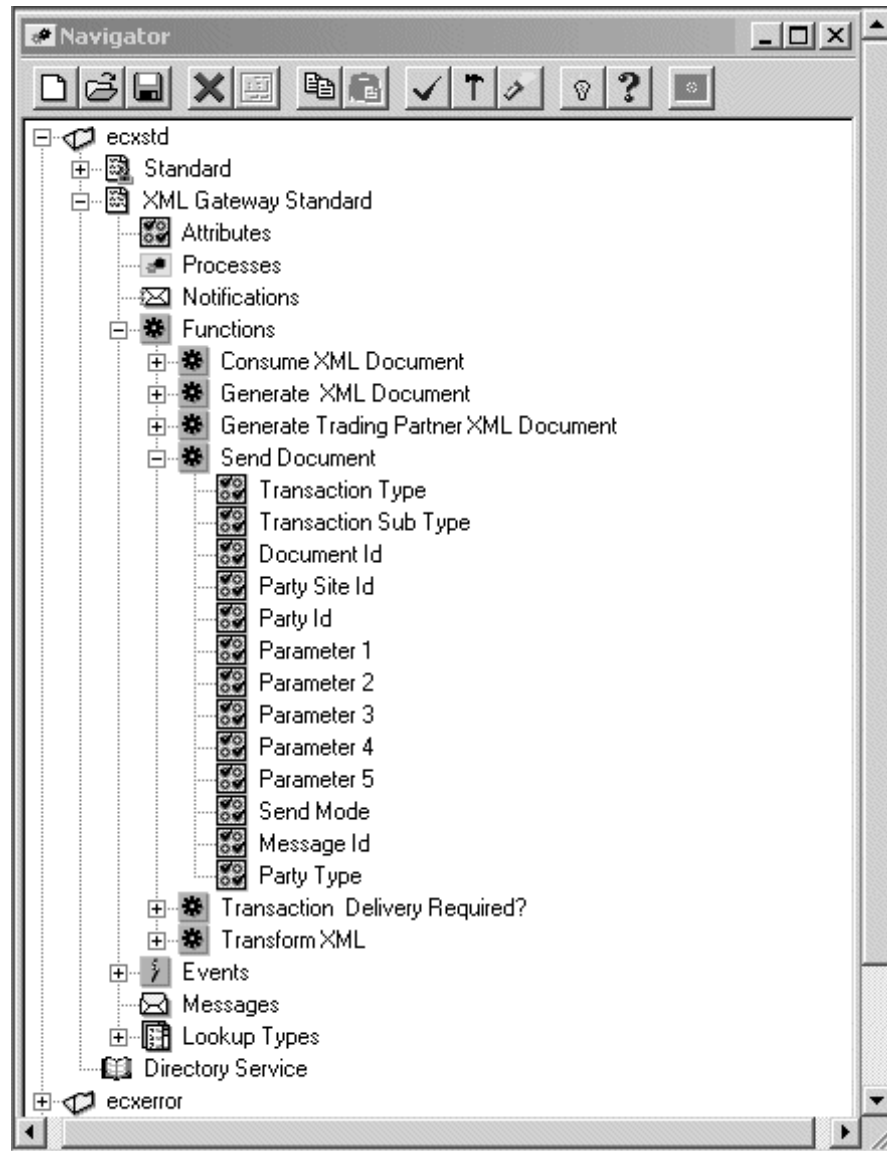
### Send Document (applies to B2B for outbound messages)

The Send Document function is used to trigger outbound message creation. The XML message may be created immediately or deferred depending on the setting of the Send Mode attribute. This function triggers XML Gateway to gather the data from the Oracle E-Business Suite, and to create and enqueue the XML message.

A Send Mode of "Deferred" defers the processing to the Workflow background engine.

A Send Mode of "Immediate" triggers the XML Gateway Execution Engine to create the XML message immediately after the data has been entered into the Oracle E-Business Suite. With this approach, a snapshot of the data is taken to avoid a race condition where the data may be changed after the time the event is raised and before the data is extracted.

The Send Document function may be modeled using the XML Gateway Rule Function or the Workflow Default Rule Function to ensure that XML messages are sent to Trading Partners enabled for the transaction.



The attributes for the Send Document function are shown in the following table. The values for the attributes are provided by the business event, another function activity, or are set as constants.

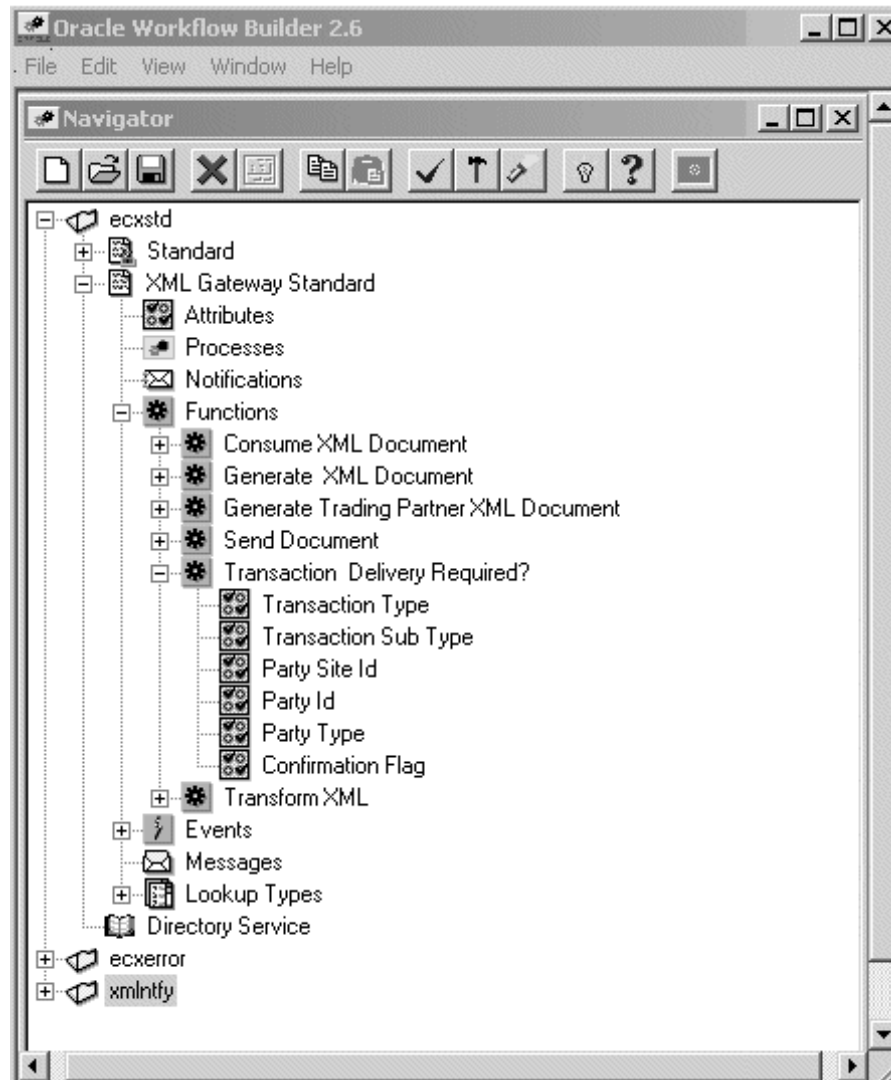
Attribute Name	Attribute Description
Transaction Type	This is the Transaction Type defined in the XML Gateway Define Transactions form.
Transaction Subtype	This is the Transaction Subtype defined in the XML Gateway Define Transactions form.
Document ID	This is the unique identifier for the business document. It can be the document number or its associated database key, whichever is guaranteed to be unique for the transaction.
Party Site ID	This is the unique identifier for the Trading Partner site defined in Oracle E-Business Suite.
Party ID	This is the unique identifier for the Trading Partner defined in Oracle E-Business Suite. This field is optional.
Parameter 1	Optional variable. This attribute is used if it is used in the message map.
Parameter 2	Optional variable. This attribute is used if it is used in the message map.
Parameter 3	Optional variable. This attribute is used if it is used in the message map.
Parameter 4	Optional variable. This attribute is used if it is used in the message map.
Parameter 5	Optional variable. This attribute is used if it is used in the message map.
Send Mode	Select "Deferred" or "Immediate"
Message ID	This is a unique identifier provided by the XML Gateway execution engine for each outbound message.
Party Type	Party Type defined in the XML Gateway Trading Partner Setup window.

### Transaction Delivery Required? (applies to B2B for outbound messages)

The Transaction Delivery Required function is used to determine if the Trading Partner is enabled for the transaction. It is used for B2B integration where Trading Partner validation is required. The function returns TRUE or FALSE.

The Transaction Delivery Required function is also embedded in the XML Gateway Rule Function. No Workflow process will be executed if the function returns FALSE.

The seeded event subscription delivered by Oracle E-Business Suite application modules can use the XML Gateway Rule Function or the Workflow Default Rule Function. The XML Gateway Rule Function is more efficient because it predetermines whether the Trading Partner is enabled for the transaction and executes the associated Workflow process only if the result of the rule function is TRUE. It is therefore the recommended approach.



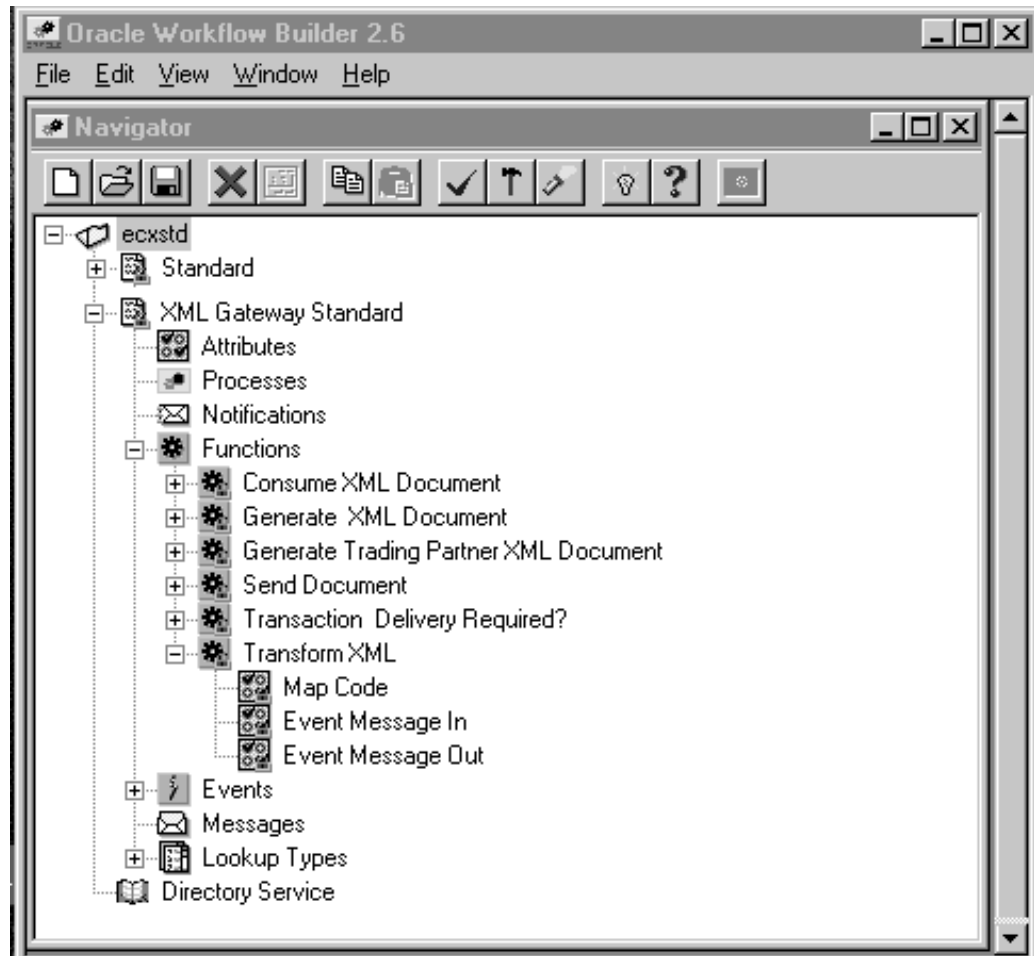
The attributes for the Transaction Delivery Required function are shown in the following table. The values for the attributes are provided by the business event, another function activity, or are set as constants.

Attribute Name	Attribute Description
Transaction Type	This is the Transaction Type defined in the XML Gateway Define Transactions form.
Transaction Subtype	This is the Transaction Subtype defined in the XML Gateway Define Transactions form.
Party Site ID	This is the unique identifier for the Trading Partner site defined in Oracle E-Business Suite.
Party ID	This is the unique identifier for the Trading Partner defined in Oracle E-Business Suite.
Party Type	The Party Type defined in the XML Gateway Trading Partner Setup window.
Confirmation Flag	The Document Confirmation flag from the Trading Partner Setup window. Indicates whether a confirmation is sent or received. The Confirmation Flag has been added to the Transaction Delivery Required? (DELIVREQUIRED) function activity. Workflows for outbound business documents can now be designed to wait for the inbound confirmation if the ECX_CONFIRMATION_FLAG is set to 1 (send confirmation if error detected) or 2 (always send confirmation). The Confirmation Flag is an attribute of the Trading Partner that is defined using the Trading Partner Setup window. Use this Workflow design approach if you want to link the outbound business document with the inbound confirmation. This is optional.

## Transform XML

The Transform XML function is used for XML to XML transformation.





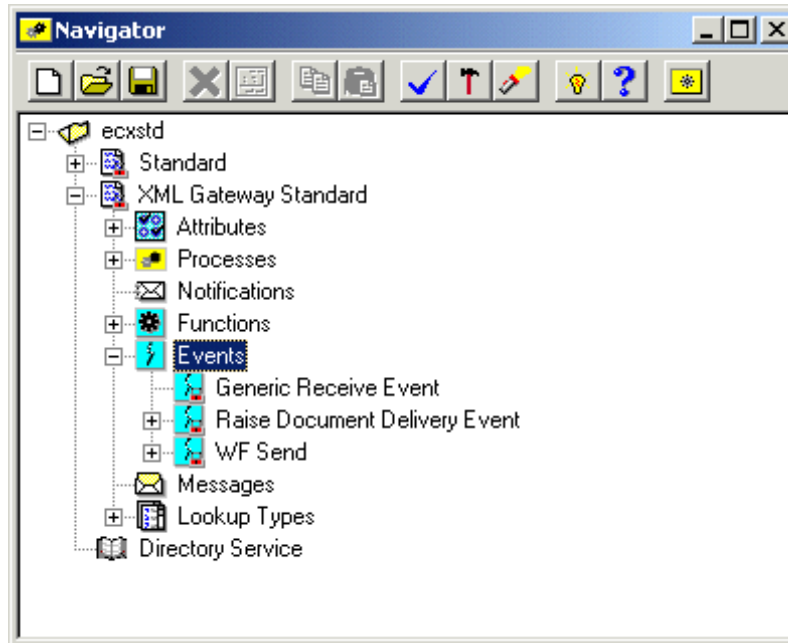
The Transform XML function transforms a given Input XML document into another XML document. The input parameters to this activity are the Map Code and the Input XML Document. The output of the activity is the transformed XML document. The execution engine will execute the transformation according to the Map Definition.

The attributes for the Transform XML function are shown in the following table. The Event Message In attribute stores the input XML information and the Event Message Out attribute stores the transformed XML information.

Attribute Name	Attribute Description
Map Code	The Map Code associated with the Transaction as defined in the Trading Partner Details form.
Event Message In	The Input XML message.
Event Message Out	The Transformed XML message.

## Events

The XML Gateway Standard Item Type includes three Events to raise a business event from an existing Workflow process.



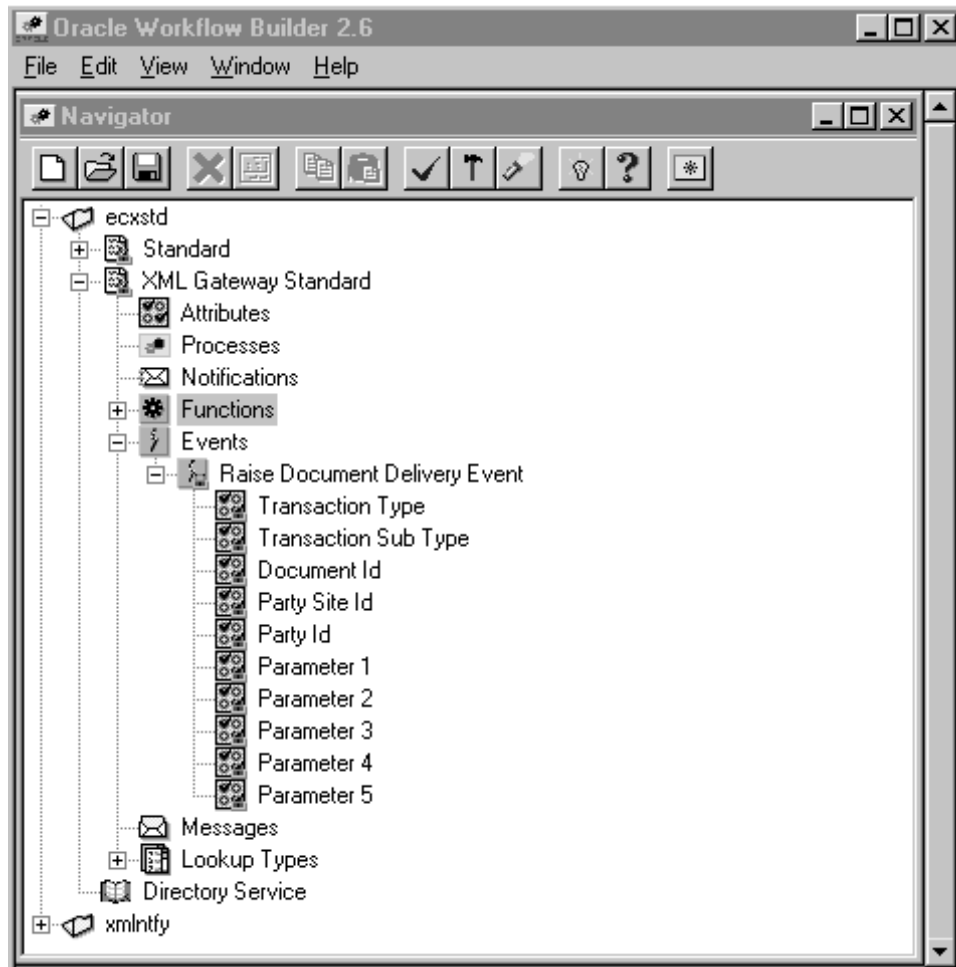
### Generic Receive Event

The Generic Receive Event is used to support the Create Trading Partner Message process.

### Raise Document Delivery Event

The Raise Document Delivery Event is used to raise a business event from an existing Workflow process. This allows you to seamlessly integrate your existing Workflow process with Oracle XML Gateway to create an outbound XML message.

An alternate approach is to raise the business event directly from PL/SQL code. Refer to the *Raise, Oracle Workflow API Reference* for the details.



The attributes for the Raise Document Delivery Event are shown in the following table. The values for the attributes are provided by a function activity or are set as constants.

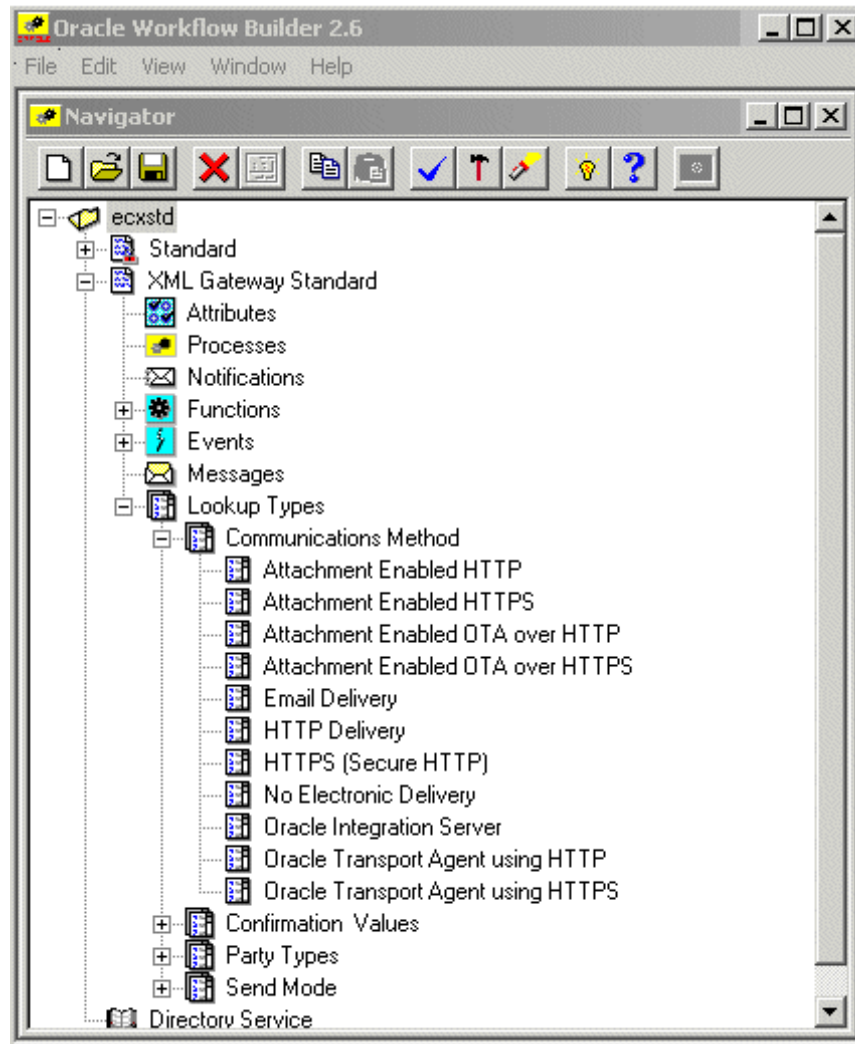
Attribute Name	Attribute Description
Transaction Type	This is the Transaction Type defined in the XML Gateway Define Transactions form.
Transaction Subtype	This is the Transaction Subtype defined in the XML Gateway Define Transactions form.
Document ID	This is the unique identifier for the business document. It can be the document number or its associated database key, whichever is guaranteed to be unique for the transaction.
Party Site ID	This is the unique identifier for the Trading Partner site defined in Oracle E-Business Suite.
Party ID	This is the unique identifier for the Trading Partner defined in Oracle E-Business Suite.
Parameter 1	Optional Variable. This attribute is used if it is used in the message map.
Parameter 2	Optional Variable. This attribute is used if it is used in the message map.
Parameter 3	Optional Variable. This attribute is used if it is used in the message map.
Parameter 4	Optional Variable. This attribute is used if it is used in the message map.
Parameter 5	Optional Variable. This attribute is used if it is used in the message map.

## WF Send

The WF Send Event enqueues a message onto the ECX\_OUTBOUND queue. This event is used in conjunction with Callback. See Message Delivery Status, page 6-55 for a complete description of this event and its attributes.

## Lookup Types

The XML Gateway Standard Item Type supports the Send Mode Lookup Type used in the Send Document function to indicate whether the event subscription should be processed immediately or in deferred mode.



### Communications Method

- Attachment Enabled HTTP
- Attachment Enabled HTTPS
- Attachment Enabled OTA over HTTP
- Attachment Enabled OTA over HTTPS
- Email Delivery
- HTTP Delivery
- HTTPS (Secure HTTP)
- No Electronic Delivery
- Oracle Integration Server
- Oracle Transport Agent Using HTTP
- Oracle Transport Agent Using HTTPS

## Confirmation Values

Confirmation Values are used in the Transaction Delivery Required function activity.

- No confirmation Business Object Document Delivery
- Send a confirmation Business Object Document Regardless
- Send back a confirmation Business Object Document only if an error has occurred

## Party Types

Party Types are used in the Get Trading Partner Role and Send Document function activities.

- Bank
- Customer
- Carrier
- Exchange
- Internal
- Supplier
- XML Gateway Standalone

## Send Mode

Send Mode is used in the Send Document function activity.

- Deferred

The Deferred mode informs the XML Gateway execution engine to defer the message creation to the Workflow background engine. This is also known as asynchronous processing.

- Immediate

The Immediate mode informs the XML Gateway execution engine to create and enqueue the XML message immediately after the data has been entered into the Oracle E-Business Suite. This is also known as synchronous processing.

Synchronous processing avoids the race condition where the data may be changed between the time the event is raised and before the data is extracted.

## XML Gateway Error Processing Item Type

The XML Gateway Error Processing Item Type contains error handling processes to manage errors detected by Oracle Workflow Business Event System or Oracle XML Gateway.

Oracle Workflow sends a notification to the Trading Partner contact for data errors or to the XML Gateway system administrator for system or process errors. For errors that require collaboration between the Trading Partner contact and XML Gateway system administrator, a notification is sent to both parties to encourage discussion and to expedite problem resolution.

The system administrator receiving a notification has the option to retry, reprocess, or abort the failed process. If the source of the error can be corrected by the system

administrator, the system administrator will correct the error and retry or reprocess the failed process.

The term "retry" is used for outbound messages. The Workflow Retry feature is used to support this requirement. The retry feature resumes from the point of failure.

The term "reprocess" is used for inbound messages. The ECX Reprocess Inbound function is used to support the reprocess feature. The function resumes using a copy of the inbound message stored in the ECX\_DOCLOGS table.

If the source of the error is in the accuracy of the message data received from the Trading Partner, or of the data setup in the Trading Partner's application, the system administrator will abort the failed process and wait for the Trading Partner to send the corrected message (in the case of an inbound message) or to correct their data setup (in the case of an outbound message).

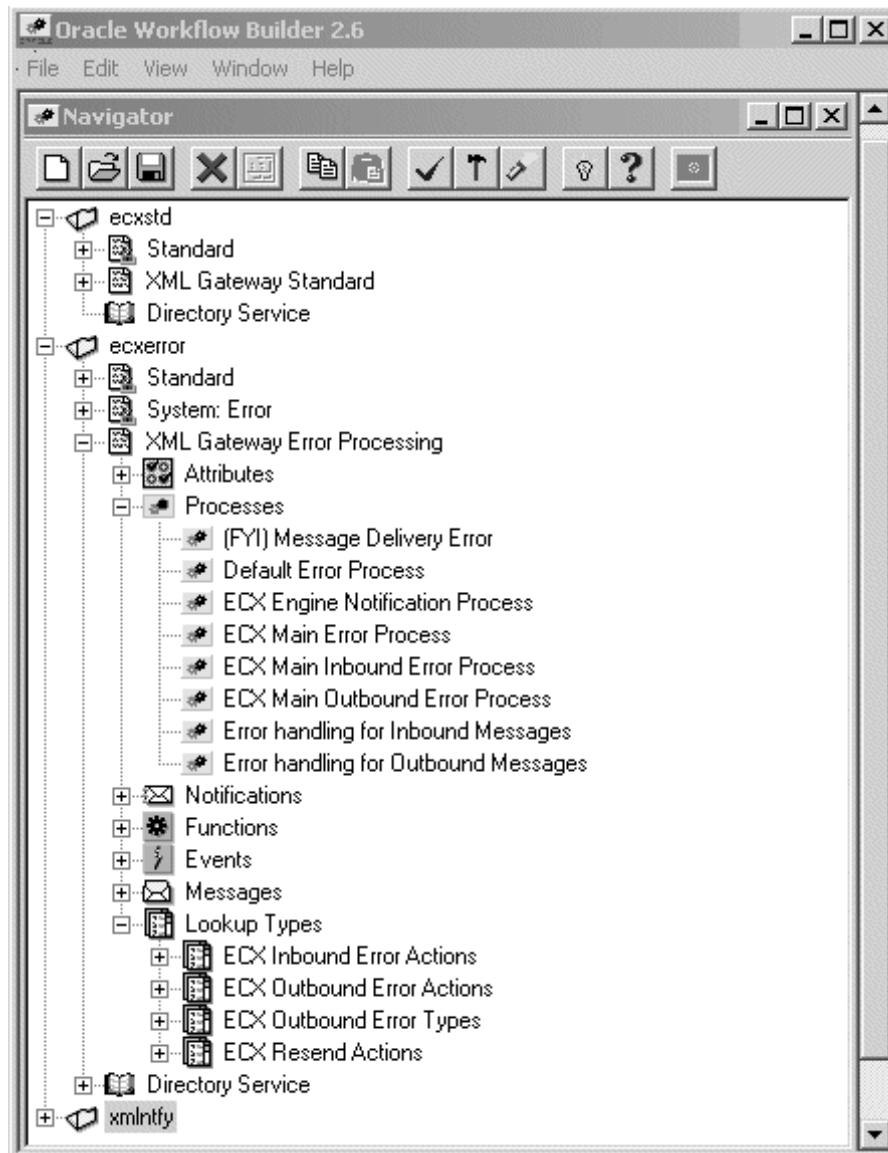
## Attributes

Attributes, also known as variables, are used to support the function, event, notification, and process activities defined for the XML Gateway Error Processing Item Type.

## Processes

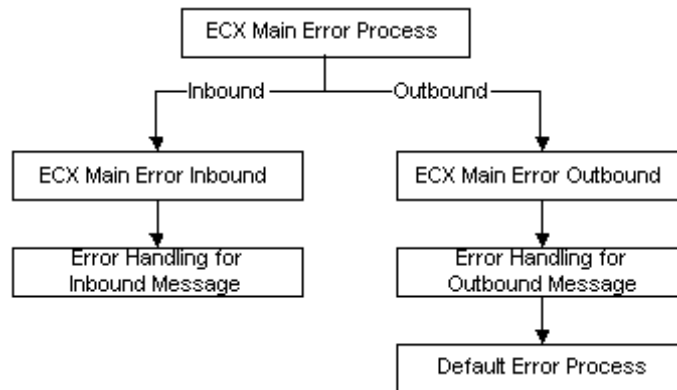
The XML Gateway Error Processing Item Type supports the following error handling processes:

- (FYI) Message Delivery Error
- Default Error Process
- ECX Engine Notification Process
- ECX Main Error Process
- ECX Main Inbound Error Process
- ECX Main Outbound Error Process
- Error handling for Inbound Messages
- Error handling for Outbound Messages



The relationship of the error handling processes is shown below. The ECX Main Error Process calls the ECX Main Error Inbound process for inbound message errors or the ECX Main Error Outbound process for outbound message errors. If inbound, the ECX Main Error Inbound process calls the Error Handling for Inbound Message process. If outbound, the ECX Main Error Outbound process calls the Error Handling for Outbound Message process, which calls the Default Error process.



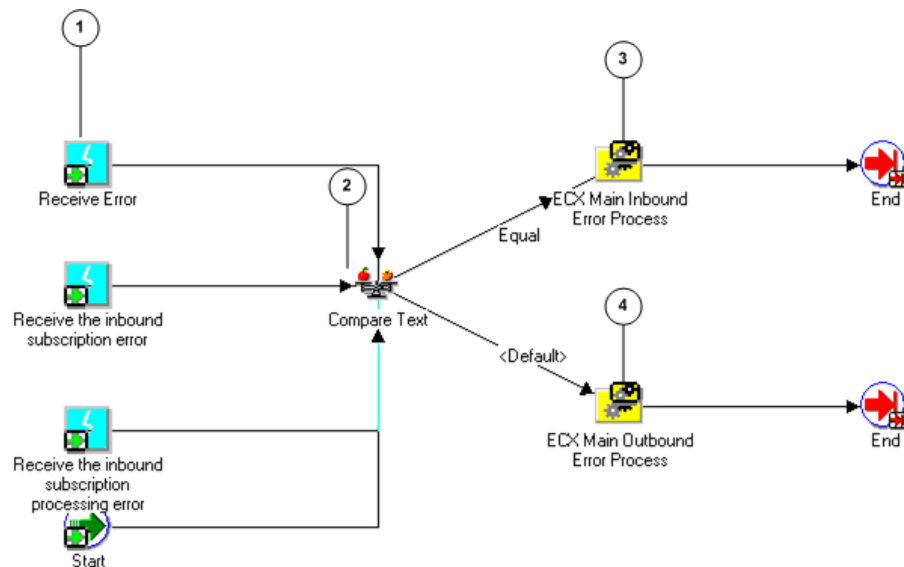


The following is a description of each Process (listed in their functional order above) supported by the XML Gateway Error Processing Item Type:

### ECX Main Error Process

The ECX Main Error Process is triggered by the Receive Error Event (see illustration, node 1), the Receive Inbound Subscription Error event, or the Receive Inbound Subscription Processing Error event for errors detected by the XML Gateway execution engine; or by Workflow for errors involving the Workflow processes created using the XML Gateway Standard Item Type.

At node 2, the process determines whether the error relates to an inbound or outbound message. The ECX Main Error Process calls the ECX Main Inbound Error Process (node 3) for errors related to an inbound message. For errors related to an outbound message, it calls the ECX Main Outbound Error Process (node 4).



### ECX Main Inbound Error Process

As shown in the illustration below, ECX Main Inbound Error Process gets the error information (node 1) from the XML Gateway execution engine or Workflow engine and calls the Error Handling for Inbound Messages process (node 2).



## Error Handling for Inbound Messages

Errors detected by the XML Gateway execution engine or Workflow engine may trigger a notification. The notification can be sent to the XML Gateway system administrator, the Trading Partner contact, or both depending on the nature of the error.

Notifications can be e-mailed to the target recipient if the Workflow mailer is enabled. The system administrator's e-mail address is defined in the ECX: System Administrator Email Address system profile. The Trading Partner contact e-mail address is defined on the Trading Partner form.

The target recipient(s) of the notification are predefined for the error message and cannot be changed.

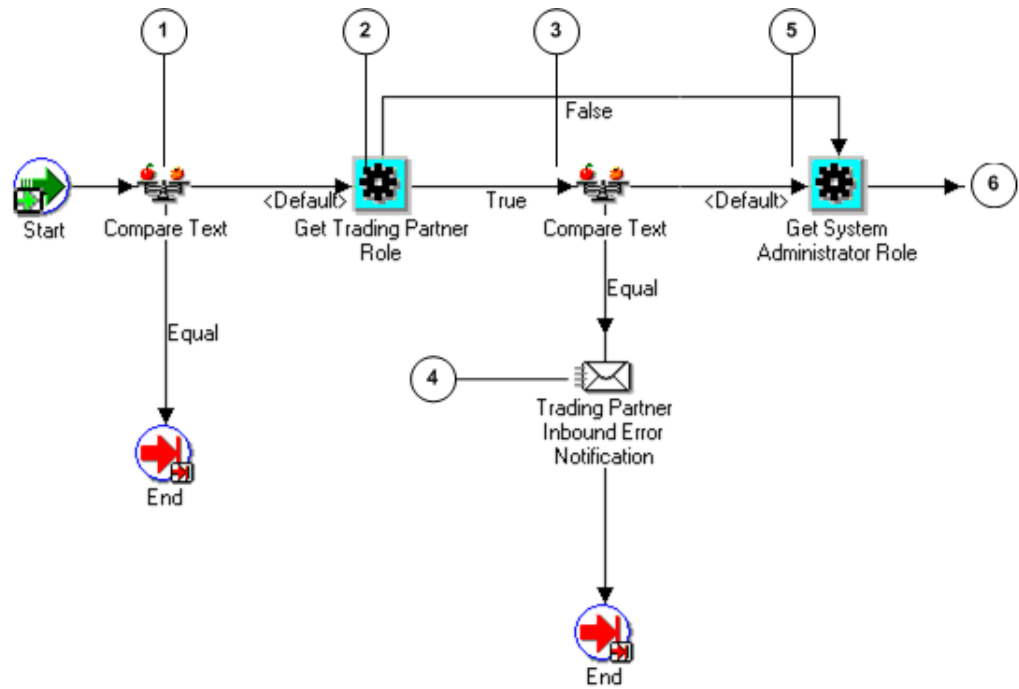
The Error Handling for Inbound Messages process checks the type of the error detected (shown in the following diagram, node 1). If the error type indicates that no notification is sent, the failed process ends, and the error process ends.

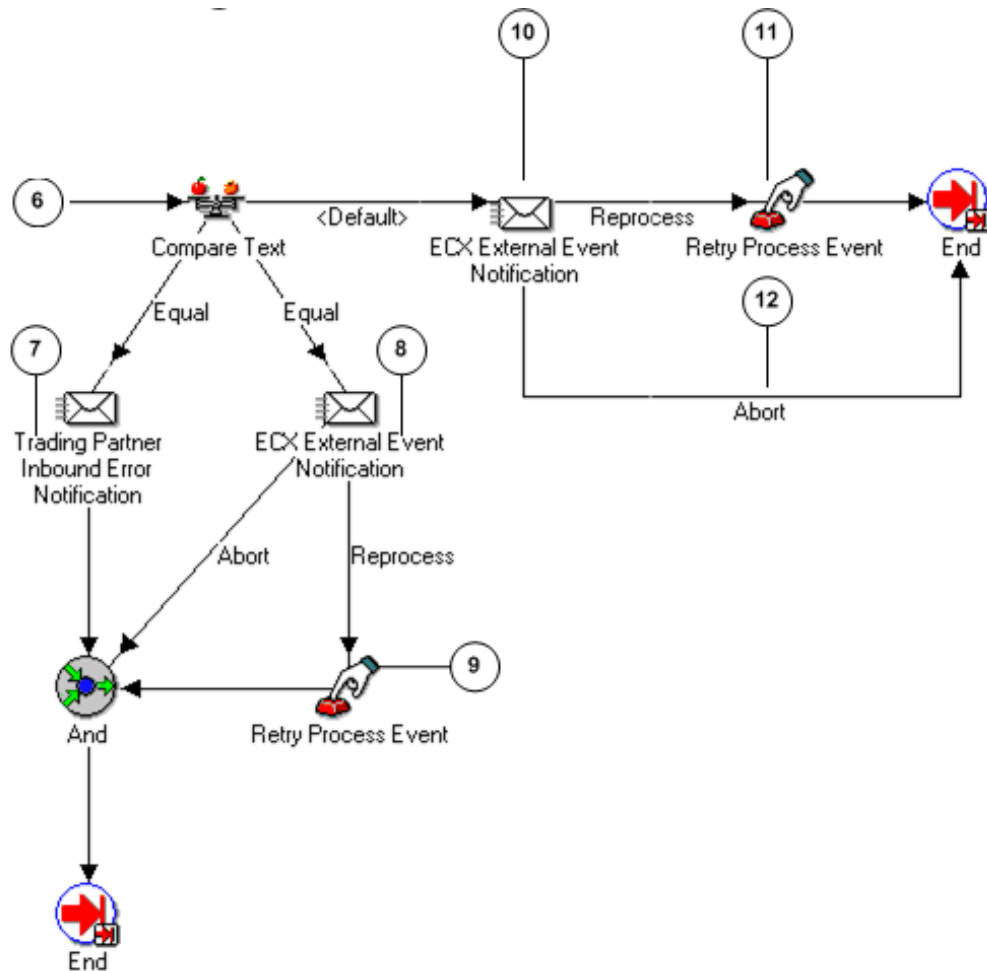
If the compare in node 1 indicates that a notification is to be sent, the process retrieves the trading partner contact information (Company Admin e-mail) from the trading partner tables (node 2). If the error type indicates that the target recipient is the Trading Partner contact (performed in node 3), a notification is sent to the Trading Partner (node 4), the failed process ends, and the error process ends.

If the check in node three did not indicate that the target recipient is solely the Trading Partner contact, the process retrieves the System Administrator contact information (node 5) and checks the error type again (node 6).

If the target recipients are both the Trading Partner contact and the XML Gateway system administrator, then two notifications are sent, one to each party (nodes 7 and 8). The system administrator has the option of retrying or aborting the process (node 9).

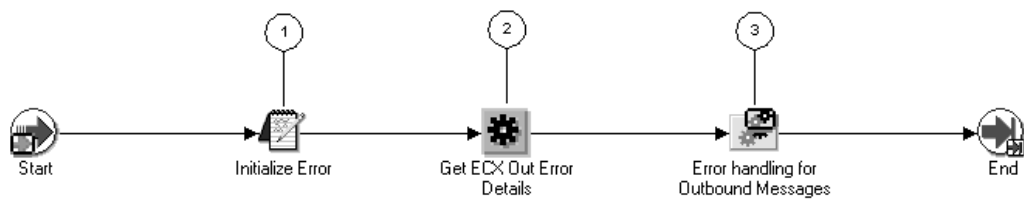
If the target recipient is solely the XML Gateway system administrator, the system administrator is notified (node 10) and has the option to reprocess (node 11) or abort (node 12) the failed inbound process. Reprocessing resumes using a copy of the inbound message stored in the ECX\_DOCLOGS table.





### ECX Main Outbound Error Process

ECX Main Outbound Error Process initializes the error (shown below, node 1) and gets the error information (node 2) from the XML Gateway execution engine or Workflow engine and calls the Error Handling for Outbound Messages process (node 3).



### Error Handling for Outbound Messages

Errors detected by the XML Gateway execution engine or Workflow engine may trigger a notification. The notification can be sent to the XML Gateway system administrator, the Trading Partner contact, or both depending on the nature of the error.

Notifications may be e-mailed to the target recipient if the Workflow mailer is enabled. The system administrator's e-mail address is defined in the ECX: System

Administrator Email Address system profile. The Trading Partner contact e-mail address is defined on the Trading Partner form.

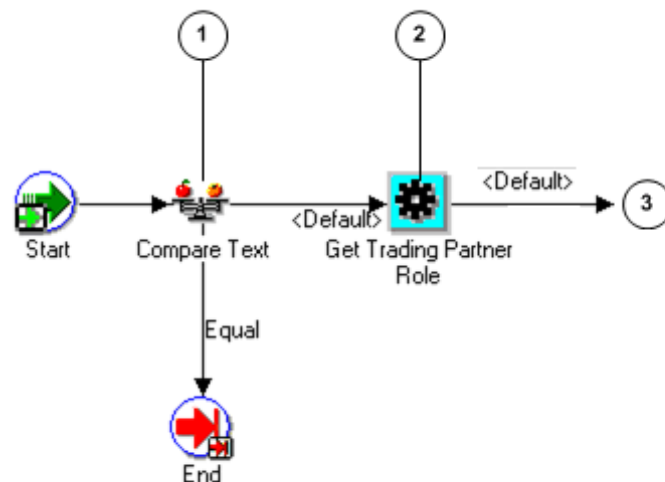
The target recipient(s) of the notification are predefined for the error message and cannot be changed.

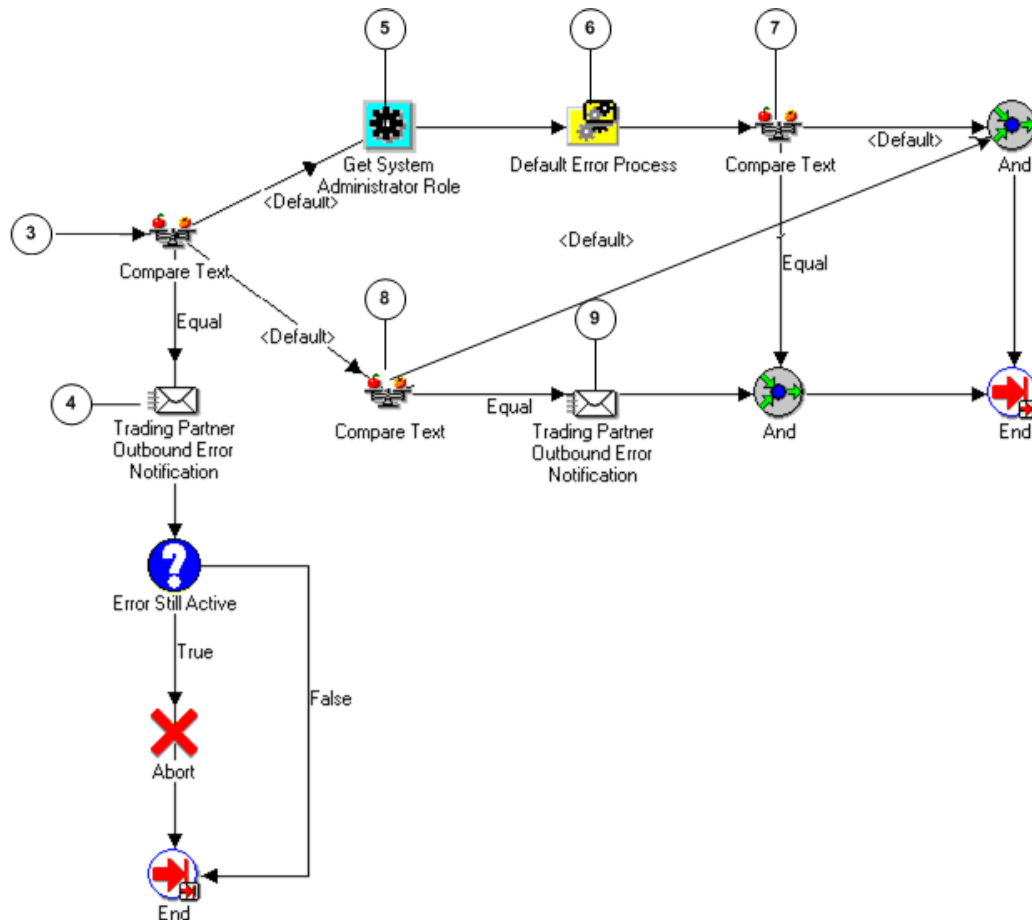
The Error Handling for Outbound Messages process checks the error type to determine if a notification is to be sent (shown below, node 1). If the type indicates that no notification is to be sent, the process ends.

If the type does not indicate that no notification is to be sent, the process retrieves the trading partner contact information (Company Admin e-mail) from the trading partner tables (node 2). The notification type is checked (node 3), and if the target recipient is solely the Trading Partner contact, a notification is sent to the Trading Partner (node 4) and the failed process is aborted.

If the target recipient is not solely the trading partner contact, the system administrator contact information is retrieved (node 5) and the Default Error Process is called (node 6). At node 8 the type is checked to determine if the notification is to be sent to both the trading partner and the system administrator.

If the target recipients are both the Trading Partner contact and the XML Gateway system administrator, then notification is also sent to the Trading Partner contact (node 9).

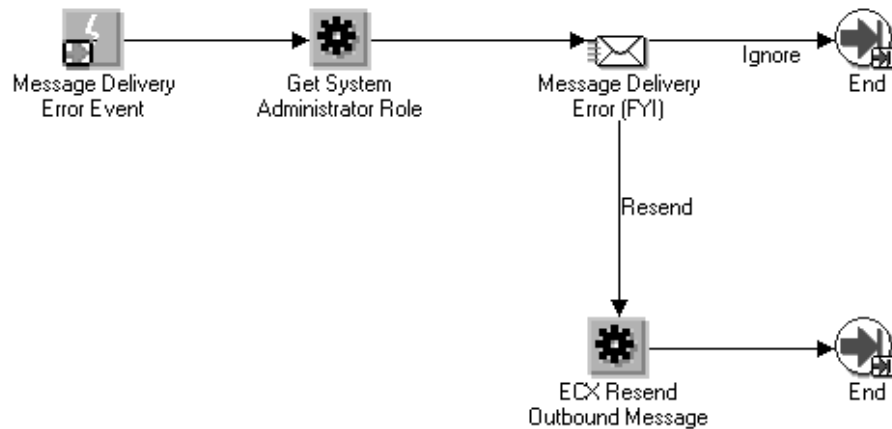




### (FYI) Message Delivery Error Process

The (FYI) Message Delivery Process is used with Oracle Transport Agent (or other messaging systems) to report message delivery status back to the Workflow process that initiated message creation.

A corresponding event subscription (seeded in XML Gateway) for the `oracle.apps.ecx.processing.message.callback` event initiates the XML Gateway Standard Error Processing Item Type: `FYI_MESSAGE_DELIVERY_ERROR` error handling process. In the event of an error, a notification is sent to the System Administrator who has the option to "Resend" the message after correcting the error, or to "Ignore" the notification if no action is required.

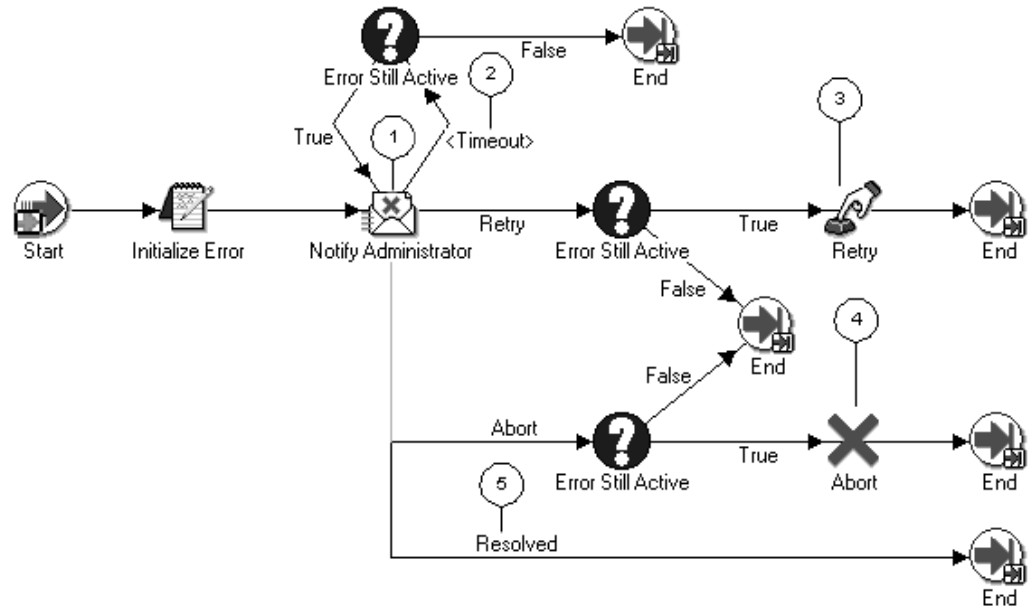


### Default Error Process

The Default Error Process is called by the Error Handling for Outbound Messages process when the error message notification is for the system administrator.

The Workflow notification administrator delivers the notification (shown below, node 1) to the system administrator and proceeds on one of the following paths:

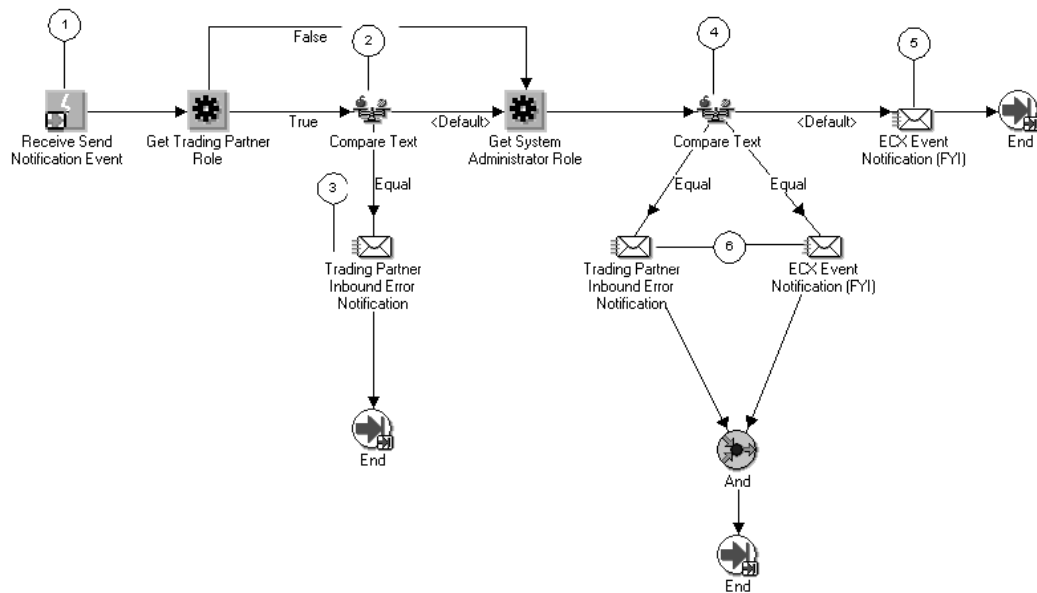
- Check for response from the system administrator and issue time-out if no response received (node 2).
- Execute retry function (node 3) if the system administrator wants to retry the failed outbound process. The retry process resumes from the point of failure. Continue retry function until error is resolved or failed process is aborted.
- Execute abort function (node 4) if the system administrator wants to abort the failed outbound process.
- Resolve error and end process (node 5)



## ECX Engine Notification Process

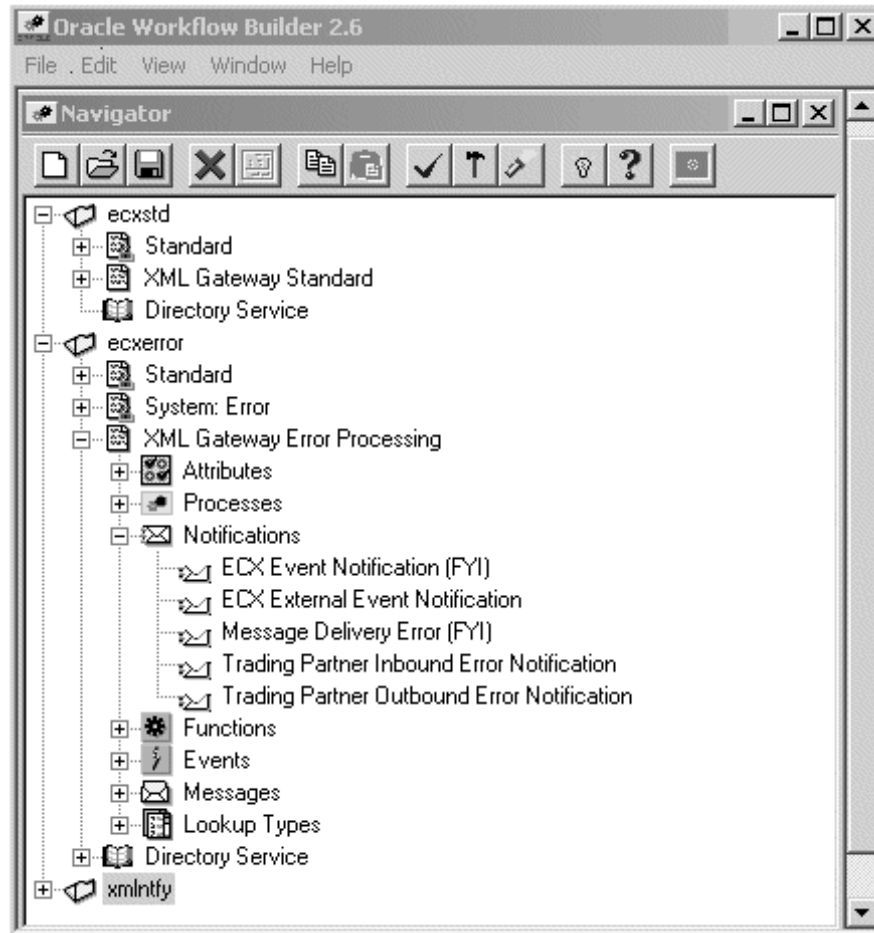
The Receive Send Notification Event (shown below, node 1) triggers the ECX Engine Notification Process.

The ECX Engine Notification Process checks the error type to determine the target recipient (node 2). If the target recipient is the Trading Partner contact, notification is sent (node 3). If the error type does not indicate that the Trading Partner contact is the sole target recipient, the process checks the error type (node 4) to determine if the notification should be sent to the XML Gateway system administrator (node 5), or both parties (node 6).





## Notifications



The following is a description of each Notification supported by the XML Gateway Error Processing Item Type:

### ECX Event Notification (FYI)

The ECX Event Notification [FYI] is used to send a notification to the system administrator.

This notification is used by the ECX Engine Notification Process.

### ECX External Event Notification

The ECX External Event Notification is used to send a notification to the system administrator.

This notification is used by the Error Handling for Inbound Messages process.

### Message Delivery Error (FYI)

The Callback feature allows messaging systems (including OTA) to report message delivery status back to the Workflow process that initiated message creation. If delivery fails, the `apps.ecx.processing.message.callback` event is raised. The corresponding event subscription for the `FYI_MESSAGE_DELIVERY_ERROR` process is executed and a

Workflow notification is sent to the System Administrator (defined in the ECX: System Administrator Email Address profile) who has the option to retry/resend or about/ignore the process.

### Trading Partner Inbound Error Notification

The Trading Partner Inbound Error Notification is used to send a notification to the Trading Partner contact.

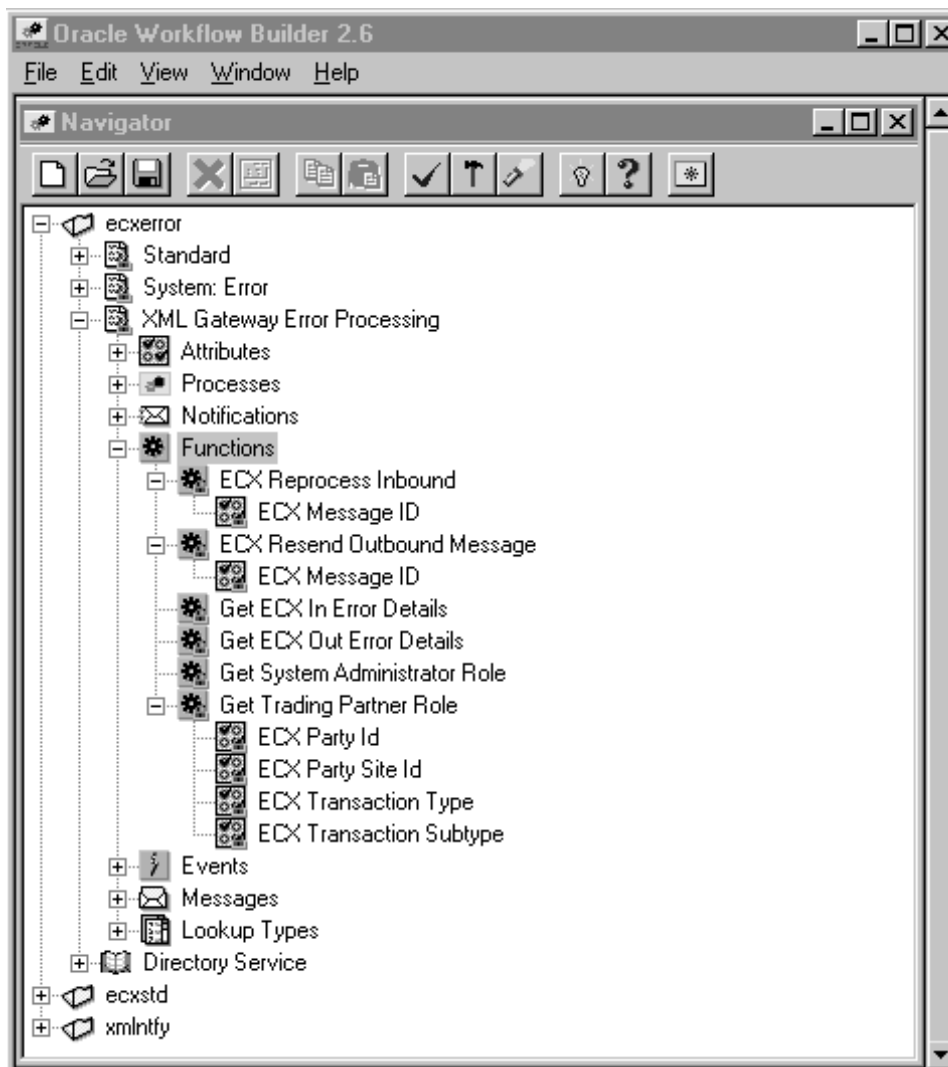
This notification is used by the Error Handling for Inbound Message and ECX Engine Notification processes.

### Trading Partner Outbound Error Notification

The Trading Partner Outbound Error Notification is used to send a notification to the Trading Partner contact.

This notification is used by the Error Handling for Outbound Message process.

## Functions



Several functions are provided in the XML Gateway Error Processing Item Type to support the error handling process.

### **ECX Reprocess Inbound**

The ECX Reprocess Inbound function is used if the XML Gateway system administrator responded to the error notification by selecting the "reprocess" option. The reprocess function will rerun the inbound process using a copy of the inbound message stored in the ECX\_DOCLOGS table.

The "reprocess" function is not supported for external notifications to the Trading Partner.

The attribute for the ECX Reprocess Inbound function is:

#### **ECX Message ID**

This is a unique identifier provided by the XML Gateway execution engine for each outbound message.

### **ECX Resend Outbound Message**

The ECX Resend Outbound function is provided to support Oracle Exchange only, it is not used in the XML Gateway Error Processing Item Type. The ECX Resend Outbound function resumes from a copy of the outbound message stored in the ECX\_DOCLOGS table.

The Oracle Workflow "retry" function is used to rerun an outbound process from the point of failure. This is handled using the Default Error Process.

The attribute for the ECX Resend Outbound function is shown in the following table:

#### **ECX Message ID**

This is a unique identifier provided by the XML Gateway execution engine for each outbound message.

### **Get ECX In Error Details**

The Get ECX In Error Details function activity is used to get the details regarding an inbound error to prepare the e-mail notification. In addition, required information is passed to the Error Handling for Inbound Messages process activity.

### **Get ECX Out Error Details**

The Get ECX Out Error Details function is used to get the details regarding an outbound error to prepare the notification. In addition, required information is passed to the Error Handling for Outbound Messages process.

### **Get System Administrator Role**

The Get System Administrator Role function is used to retrieve the e-mail address for the system administrator stored in the ECX: System Administrator Email Address system profile.

The e-mail address retrieved from the system profile is returned by the function and stored in the ECX System Administrator Role (ECX\_SA\_ROLE) item attribute.

A notification is sent to the XML Gateway system administrator for system or process errors detected by Oracle XML Gateway or Oracle Workflow Business Event System.

## Get Trading Partner Role

The Get Trading Partner Role function is used to determine the e-mail address for the Trading Partner contact that was provided when the Trading Partner was defined.

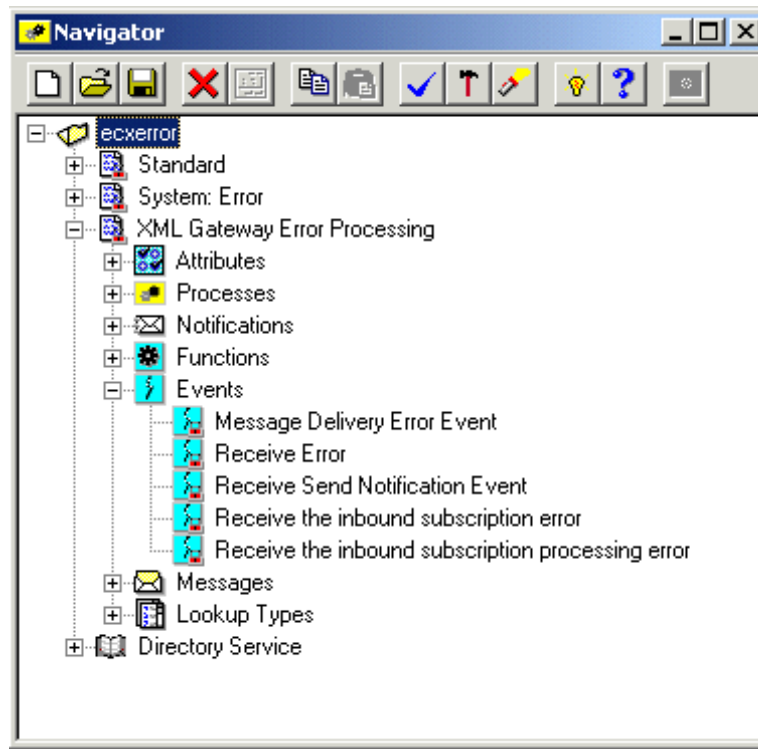
The function uses the attribute values passed to it to uniquely identify the Trading Partner. The e-mail address associated with the Trading Partner selected returned by the function is stored in the ECX Trading Partner Role (ECX\_TP\_ROLE) item attribute.

A notification is sent to the Trading Partner contact for data errors detected by Oracle XML Gateway.

The attributes for the Get Trading Partner Role function are shown in the following table:

Attribute Name	Attribute Description
ECX Party ID	This is the unique identifier for the Trading Partner defined in Oracle E-Business Suite. This field is optional.
ECX Party Site ID	This is the unique identifier for the Trading Partner site defined in Oracle E-Business Suite.
ECX Transaction Type	This is the Transaction Type defined in the XML Gateway Define Transactions form.
ECX Transaction Sub Type	This is the Transaction Sub Type defined in the XML Gateway Define Transactions form.

## Events



The XML Gateway Error Processing Item Type supports the following event activities:

- Message Delivery Error

- Receive Error
- Receive Send Notification Event
- Receive the inbound subscription error
- Receive the inbound subscription processing error

All of the XML Gateway seeded events and event subscriptions are seeded with a Customization Level of "L" for Limit, which means the event or event subscription may be enabled or disabled but may not be changed in any other way. Because the seeded events and event subscriptions play an integral role in the total solution, Oracle does not recommend that you disable any of them. If necessary, you may define additional events and event subscriptions to augment the seeded activities.

### **Message Delivery Error**

The Message Delivery Error Event is used by the XML Gateway to indicate that an error has occurred. This event is used with OTA Callback.

The seeded event name is `oracle.apps.ecx.processing.message.callback`. This event triggers the (FYI) Message Delivery Error Process workflow.

### **Receive Error**

The Receive Error event is used by XML Gateway to indicate that the execution engine has detected an error. Regardless of whether the error was detected by Oracle XML Gateway or Oracle Workflow Business Event System, the same ECX Main Error Process manages the error. See: ECX Main Error Process, page 6-27 for the details.

The seeded event name is `oracle.apps.ecx.processing.message.error`. This event triggers the ECX Main Error Process workflow.

### **Receive Send Notification Event**

The Receive Send Notification Event is used by XML Gateway to indicate that the execution engine has identified a need to send a notification for errors related to an inbound process. The ECX Engine Notification Process manages the error. See: ECX Engine Notification Process, page 6-34 for the details.

The seeded event name is `oracle.apps.ecx.processing.notification.send`. This event triggers the ECX Engine Notification Process workflow.

### **Receive the inbound subscription error**

The event `"oracle.apps.ecx.inbound.message.receive"` is raised by the ECX\_INBOUND agent listener after it dequeues a message from the ECX\_INBOUND queue. The "Receive Inbound Subscription Error" Event is used to indicate that there is an error when executing the subscription for the event `"oracle.apps.ecx.inbound.message.receive"`.

This subscription saves the xml message to `ecx_doclogs`, checks the trading partner setup for the given transaction, performs logging, and enqueues the message to the next queue for the actual map processing.

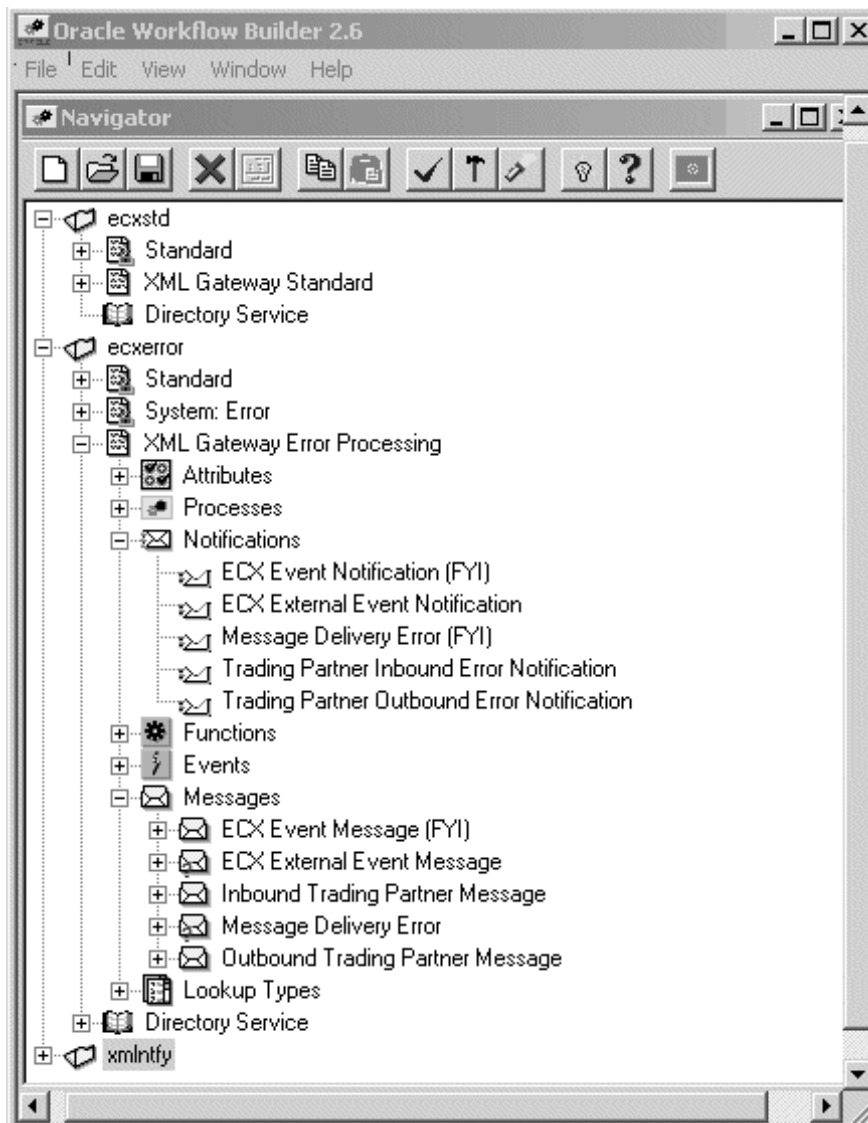
### **Receive the inbound subscription processing error**

The event `"oracle.apps.ecx.inbound.message.process"` is raised by the ECX\_TRANSACTION agent listener after it dequeues a message from the

ECX\_IN\_OAG\_Q queue. The "Receive Inbound Subscription Processing Error" is used to indicate that there is an error when processing the inbound xml document.

## Messages

The XML Gateway Error Processing Item Type provides several message templates used by the various notification activities to send a notification to the XML Gateway system administrator or the Trading Partner contact based on the nature of the error.



Below is a description of each message template. The message template is displayed in the Body tab and Text Body subtab.

### ECX Event Message (FYI)

The ECX Event Message [FYI] template is used by the ECX Event Notification. The template includes information regarding the transaction, the type of error detected, and the associated error message.

## ECX External Event Message

The ECX External Event Message template is used by the ECX External Event Notification activity. The template includes information regarding the transaction, the type of error detected, and the associated error message.

## Inbound Trading Partner Message

The Inbound Trading Partner Message template is used by the Trading Partner Inbound Error Notification activity. The template includes information regarding the transaction, the type of error detected, and the associated error message.

## Message Delivery Error

The Message Delivery Error template is used by the XML Message Delivery Callback error process. The template includes information regarding the action to take in case of error.

## Outbound Trading Partner Message

The Outbound Trading Partner Message template is used by the Trading Partner Outbound Error Notification activity. The template includes information regarding the transaction, the type of error detected, and the associated error message.

## Message Template Attributes

Each message template has the attributes listed in the following table. The values for the attributes are provided by the Get ECX Out Error Details and Get ECX In Error Details function. The values are used to replace the message tokens in the message template.

For a detailed explanation of each error and possible corrective actions, see Manual Troubleshooting Steps, page G-11.

The following table describes all the possible attributes:

- Event Key is a unique identifier for an instance of an event. The combination of event name, event key, and event data fully describe what occurred in the event.

Attribute Name	Attribute Description
Event Key	Event Key is a unique identifier for an instance of an event. The combination of event name, event key, and event data fully describe what occurred in the event.
ECX Transaction Type	This is the Transaction Type defined in the XML Gateway Define Transactions form.
ECX Document ID	This is the unique identifier for the business document. It can be the document number or its associated database key, whichever is guaranteed to be unique for the transaction.
ECX Error Message	This is the error message text describing the error detected.

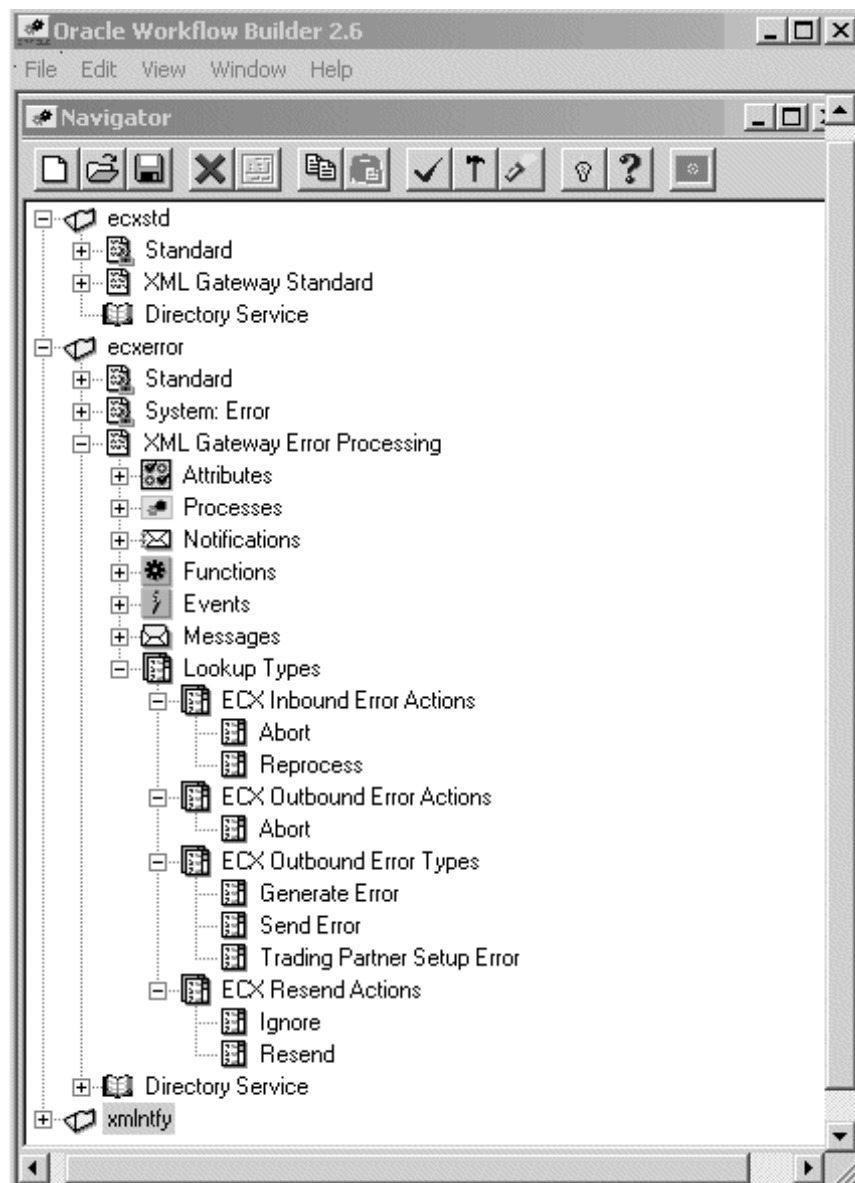
Attribute Name	Attribute Description
ECX Error Type	<p>This is the error type code. The valid values are:</p> <ul style="list-style-type: none"> <li>• 10 = Success</li> <li>• 20 = Send error notification to Trading Partner contact</li> <li>• 25 = Send error notification to both Trading Partner contact and XML Gateway System Administrator</li> <li>• 30 = Send error notification to XML Gateway System Administrator</li> </ul>
ECX Return Code	<p>This is the return status code. The valid values are:</p> <ul style="list-style-type: none"> <li>• 0 = Success</li> <li>• 1 = Warning</li> <li>• 2 = Error</li> </ul>
ECX Message Standard	Message format standard (such as OAG) as defined in the Define Transactions form.
ECX TP Header ID	Trading Partner name.
ECX Message Type	Message Type, which is XML.
ECX Logfile	If a log file is created, the name and location.
ECX Status	The status of the transaction.
ECX Time Stamp	The date the log file was created.
ECX Transaction Subtype	This is the Transaction Subtype defined in the XML Gateway Define Transactions form.
ECX Party ID	This is the unique identifier for the Trading Partner defined in Oracle E-Business Suite.
ECX Party Site ID	This is the unique identifier for the Trading Partner site defined in the XML Gateway Define Transactions form.
ECX Internal Control Number	The Internal Control Number is a system-generated number that uniquely identifies the message being processed.
Event Name	This is a unique identifier for the business event. The naming convention is oracle.apps.<product code>.<component>.<object>.<event> The Event Name is required for the Generate XML Document function to store the value returned.
ECX Party Admin Email	System Administrator as defined in the Trading Partner Setup window.
ECX Protocol Address	Protocol address used (for example, SMTP, HTTP, HTTPS).
ECX Attribute 1	Optional variable as defined in the message map.
ECX Attribute 2	Optional variable as defined in the message map.
ECX Attribute 3	Optional variable as defined in the message map.
ECX Attribute 4	Optional variable as defined in the message map.



Attribute Name	Attribute Description
ECX Attribute 5	Optional variable as defined in the message map.
ECX Trigger ID	Unique number given to each event being raised.
ECX Trading Partner Role	The e-mail address from the Trading Partner Setup window that is returned by the Get Trading Partner Role function.
ECX Protocol Type	Transmission protocol as defined in the Setup Trading Partner window.

## Lookup Types

The XML Gateway Error Processing Item Type supports several lookup types for the error handling processes.



## ECX Inbound Error Actions

The ECX Inbound Error Actions lookups are used by the ECX External Event Notification as a Result Type attribute. The valid values are as follows:

- Abort  
Abort the failed inbound process.
- Reprocess  
Reprocess the failed inbound process.

## ECX Outbound Error Actions

The ECX Outbound Error Actions lookups is used by the Default Error Process. The valid value is as follows:

- Abort  
Abort the failed outbound process
- The retry function is supported by the Default Error Process.

## ECX Outbound Error Types

The ECX Outbound Error Types lookups are used by the Get ECX Error Type function as a Result Type attribute. The valid values are as follows:

- Generate Error  
The error was detected during the message creation process as opposed to the message delivery process.
- Send Error  
The error was detected during the message delivery process as opposed to the message creation process.
- Trading Partner Setup Error  
The error was based on incorrect Trading Partner setup. Correct the erroneous set up before proceeding further.

## ECX Resend Actions

ECX Resend Actions are lookups used by the XML Gateway Error Process. The valid values are as follows:

- Ignore
- Resend

# Configure Oracle Prebuilt Inbound Messages

## Seeded Business Event and Corresponding Event Subscription

Inbound messages that are prebuilt and delivered by Oracle E-Business Suite application modules are delivered with the following:

- Message map containing an XML Gateway Procedure Call action as the last activity. This activity sets event details to indicate that XML Gateway has successfully processed the inbound message.
- Seeded event subscription to consume the business event raised in the message map.  
The event subscription is seeded as an enabled subscription. You can disable the event subscription if necessary.

An event subscription is required for inbound messages from Trading Partners (B2B) regardless of whether the application is interested in the inbound message. The seeded event subscription uses the Workflow Default Rule Function but may be configured to use a rule function and Workflow process relevant to the business requirement.

See the Message Designer, page 2-1 for instructions on how to create a new message map and how to modify an Oracle prebuilt message map.

## Configuration Options for Seeded Event Subscription

Use the Workflow Administrator to configure the seeded event subscription for the prebuilt inbound message if necessary.

### 1. Register new event subscription

Use the Workflow Administrator Add Event Subscription window to register the new event subscription if you defined a new subscription instead of configuring the seeded subscription.

See Manage Workflow Processes, page 6-49 for instructions on how to register new event subscriptions.

### 2. Configure Seeded Subscription

Use Oracle Workflow Administrator Add Event Subscription window to configure the seeded event subscription to consume the inbound message if your application is interested in it. The subscription will trigger the Oracle E-Business Suite to perform some activity. The actual activity performed by the event subscription will be based on the Workflow process defined for it. Some examples are as follows:

- Integrate with existing Workflow process defined in the Oracle E-Business Suite
- Call an application API to perform a specific function

See: Manage Workflow Processes, page 6-49 for instructions on how to register new event subscriptions and configure a seeded event subscription.

## Configure Oracle Prebuilt Outbound Messages

### Seeded Business Event and Corresponding Event Subscription

Outbound messages that are prebuilt and delivered by Oracle E-Business Suite application modules are delivered with the following:

- Calls in the application module to raise a business event to indicate when something of interest has occurred. This includes event points indicating when a document was created, changed, confirmed, or deleted.
- Seeded event subscription using the XML Gateway or Workflow Default Rule Function to consume all the business events raised in the application module.

The event subscription is seeded as an enabled subscription. You can disable the event subscription if necessary.

The seeded event subscription consumes all the seeded events but may be configured to execute a specific Workflow process for the outbound events it is interested in.

See the Message Designer chapter, page 2-1 for instructions on how to create a new message map and how to modify an Oracle prebuilt message map.

## Configuration Options for Seeded Event Subscription

Use the Workflow Administrator to configure the seeded event subscription for the prebuilt outbound message if necessary.

### 1. Register new event subscription

Use the Workflow Administrator Add Event Subscription window to register the new event subscription if you defined a new subscription instead of configuring the seeded subscription.

See Manage Workflow Processes, page 6-49 for instructions on how to register new event subscriptions.

### 2. Configure Seeded Event Subscription

Use Oracle Workflow Administrator Add Event Subscription window to configure the seeded event subscription to consume the application business event of interest. The subscription will trigger the Oracle E-Business Suite to perform some activity. The actual activities performed by the event subscription will be based on the Workflow process defined for it.

There are three subscription options for the Business-to-Business (B2B) integration scenario described below. The function activities outlined are supported by the XML Gateway Standard Item Type.

#### Send Document Function Using XML Gateway Rule Function

This option uses the XML Gateway Rule Function which has the Transaction Delivery Required function embedded in it to determine if the Trading Partner is enabled for the transaction before the message is generated.

The "Send Document" function can be configured to generate the message in synchronous or asynchronous mode by setting the Send Mode attribute to "Immediate" or "Deferred" respectively.

The following is an example of a Workflow process launched by the XML Gateway Rule Function. The Receive activity is followed by the Send Document function activity, which proceeds to End. The Transaction Delivery Required Function is not modeled in the Workflow process since it is already included in the rule function.



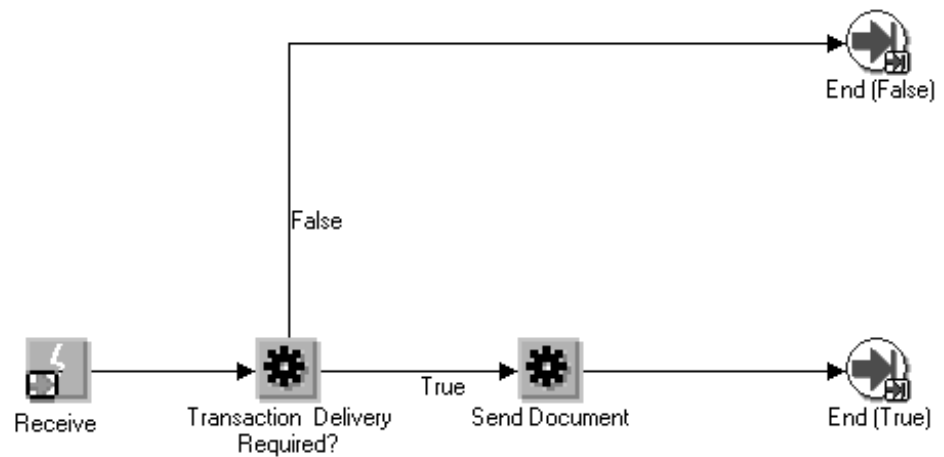
**Note:** The first Workflow activity may be a Receive or a Start activity. Use Receive to continue an existing Workflow process. Use Start to begin a new Workflow process.

### 3. Send Document Function Using Workflow Default Rule Function

This option is used to first determine if the Trading Partner is enabled for the transaction. If the result of the Transaction Delivery Required function activity returns TRUE, the "Send Document" function activity is executed.

The "Send Document" function activity may be configured to send the message in synchronous or asynchronous mode by setting the Send Mode attribute to "Immediate" or "Deferred" respectively.

The following is an example of a Workflow process launched by the Workflow Default Rule Function. The Transaction Delivery Required Function is modeled in the Workflow process since it is not included in the Workflow Default Rule Function.

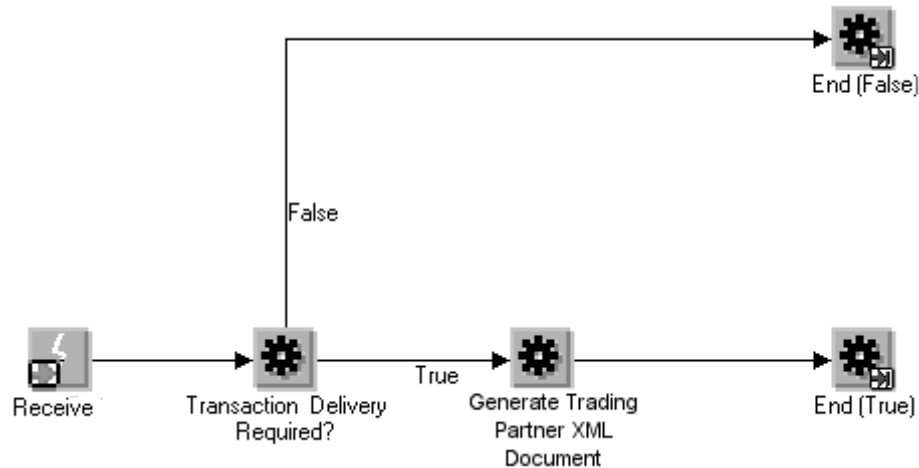


**Note:** The first Workflow step may be a Receive or a Start activity. Use Receive to continue an existing Workflow process. Use Start to begin a new Workflow process.

### 4. Generate Trading Partner XML Document

This option is used to gather and return the message data in the Event Message attribute without delivering it to the Trading Partner. The Event Message is then processed according to the subsequent Workflow instruction.

The following is an example of a Workflow process using the Generate Trading Partner XML Document function. A Receive activity is followed by the Transaction Delivery Required function activity. If it returns False, the process proceeds to End. If True, the Generate Trading Partner XML Document function activity is executed and proceeds to End. In a production Workflow process, the Generate Trading Partner XML Document function activity would be followed by an application-specific activity.



**Note:** The first Workflow step may be a Receive or a Start activity. Use Receive to continue an existing Workflow process. Use Start to begin a new Workflow process.

## Application to Application Integration

The XML Gateway Standard Item Type provides function activities to support A2A integration requirements. Listed below are the options for inbound and outbound transactions.

### Inbound Option

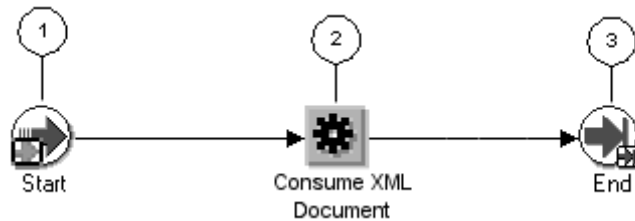
#### Consume XML Document

This option is used to interact with Oracle XML Gateway to insert data into the base application tables. Depending on how the message map was defined, the data may be inserted using an Application Open Interface API or an Application API. See: How to Map to an API, page 2-86.

The Consume XML Document function is used for A2A integration. It may be used in existing Workflow processes for which the Event Message attribute is used to pass the message data. This makes it unnecessary to raise a business event or define an event subscription as recommended in the B2B integration scenario.

The following is an example of a Workflow process using the Consume XML Document function:

This Workflow process shows the Consume XML Document (2) function activity completing and followed by an End function (3). In a production Workflow process, the Consume XML Document function activity would be followed by an application-specific activity.



**Note:** The first Workflow step (1) may be a Receive or a Start activity. Use Receive to continue an existing Workflow process. Use Start to begin a new Workflow process.

## Outbound Option

### Generate XML Message

This option is used to generate the XML message and forward it to the subsequent Workflow process. The Generate XML Document function activity is used to gather the document data and return it in the Event Message attribute. The Event Message is then processed according to the Workflow instructions which may be to send it to another Oracle E-Business Suite application module.

The following is an example of a Workflow process using the Generate XML Document function:



The Workflow process shows the Generate XML Document function activity completing and followed by an End function. In a production Workflow process, the Generate XML Document function activity would be followed by an application specific activity.

**Note:** The first Workflow step may be a Receive or a Start activity. Use Receive to continue an existing Workflow process. Use Start to begin a new Workflow process.

## Manage Workflow Processes

Use the following Oracle Workflow Administrator windows to help manage your Workflow processes.

For more information about these windows see the *Oracle Workflow Developer's Guide* and *Oracle Workflow User's Guide*.

## Register New Business Events and Event Subscription

This process is necessary for new events and subscriptions added as a result of creating a new XML message. The seeded business events and event subscriptions delivered by the Oracle E-Business Suite have already been registered. There is no need to register a seeded event subscription that was configured during implementation.

Use the Events window, *Oracle Workflow Developer's Guide* to add a new business event.

Use the Subscriptions window, *Oracle Workflow Developer's Guide* to add a new subscription to consume the new event.

## Identify Seeded Item Types

Use the Item Type Definition window, *Oracle Workflow Developer's Guide* to view the seeded item types delivered by Oracle Workflow Business Event System, Oracle XML Gateway, and Oracle E-Business Suite modules.

## Identify Seeded Business Events and Associated Event Subscriptions

Use the Events window, *Oracle Workflow Developer's Guide* to view the seeded business events delivered by Oracle Workflow Business Event System, Oracle XML Gateway, and Oracle E-Business Suite modules.

Select the business event and click on Edit Subscription to proceed to the Subscriptions window, *Oracle Workflow Developer's Guide* to view the associated subscriptions.

## Configure or Delete Seeded Event Subscriptions

From the Subscriptions window, *Oracle Workflow Developer's Guide*, select the subscription and click Edit to configure the subscription. Click Delete to delete the subscription. You can modify the rule function or Workflow process defined for the subscription.

## View and Respond to Error Notifications

Use the Worklist window, *Oracle Workflow User's Guide* to view and respond to notifications awaiting your attention.

## Purge XML Gateway Transactions

To purge workflow or non-workflow related XML Gateway transaction data, Oracle XML Gateway allows you to use the following concurrent programs to purge XML Gateway transactions:

- Purge Obsolete Workflow Runtime Data Concurrent Program, page 6-50
- Purge Obsolete ECX Data Concurrent Program, page 6-51

### Purge Obsolete Workflow Runtime Data Concurrent Program

If you are using the version of Oracle Workflow embedded in Oracle Applications, you can submit the Purge Obsolete Workflow Runtime Data concurrent program to purge XML Gateway transactions attached to workflow.

**Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to submit and manage the Purge Obsolete Workflow Runtime Data concurrent program. For more information, see the Oracle Applications Manager online help.



## Purge Obsolete ECX Data Concurrent Program

Since Workflow purge is associated with item\_type and item\_key, XML Gateway logs do not have these parameters available all the time. In case of inbound transaction item\_type and item\_key parameters are not available, you can submit the Purge Obsolete ECX Data concurrent program to specifically purge XML Gateway transaction data based on transaction type, transaction subtype, and date range.

### ***Parameters and Purge Behavior***

Parameters	Expected Behavior
Transaction Type	All data corresponding to that specific Transaction Type will be deleted.
Transaction Type Transaction Subtype	All data corresponding to that specific Transaction Type and Subtype will be deleted.
Transaction Type Transaction Subtype From Date	All data corresponding to that specific Transaction Type and Subtype will be deleted after the specified date.
Transaction Type Transaction Subtype From Date To Date	All data corresponding to that specific Transaction Type and Subtype will be deleted within the specified date range.
From Date	All data after the specified date will be deleted regardless of Transaction Type and Subtype.
From Date To Date	All data within the specified date range will be deleted regardless of Transaction Type and Subtype.
No parameter is given	All data will be deleted regardless of Transaction Type, Subtype, and date range.

Use the Submit Requests form in Oracle Applications to submit these concurrent programs.

**Note:** Commit Cycle will be of 500 rows as it is done in Workflow Purge.

## Monitor Workflow Processes

Identified below are several Oracle Workflow Administrator windows to help monitor your Workflow processes:

### Transaction-Level Trace

The trace function is performed via centralized logging through the Applications Logging Framework. Using the logging framework you can enable logging and set log levels using FND profile options. For information on this feature, see the *Oracle Applications Supportability Guide*.

## Monitor Transaction Status

Use the Event Queue Summary window to monitor the status of a transaction. Select the Agent Listener and click the View Detail icon. Once you enter the selection criteria to identify the transaction, the detail information including status and an option to view the event data in XML or text format, are presented.

## Review XML Message Returned by Generate Functions

Modify the subscription for the Generate XML Document or Generate Trading Partner XML Document function and set the Action: Out Agent field to the WF\_OUT. The XML data gathered by the function is directed to the WF\_OUT agent.

For instructions on viewing the XML message directed to WF\_OUT see: Monitor Transaction Status.

## Start Agent Listeners

There is one Agent Listener per inbound agent. The agents necessary to support the integration between Oracle Workflow Business Event System and Oracle XML Gateway are shown in the following table:

Agent Name	Description
ECX_INBOUND	Agent for inbound messages originating from outside the enterprise for B2B integration.
ECX_TRANSACTION	Agent for inbound messages originating from within the enterprise for A2A integration.
WF_OUT	Agent to stage outbound messages. ECX_OUTBOUND is the agent used to stage and deliver outbound XML messages to the Trading Partner.
WF_IN	Agent to receive inbound messages sent to the Oracle E-Business Suite.
WF_DEFERRED	Agent for outbound messages sent in deferred mode.
WF_ERROR	Agent for errors detected by Oracle Workflow Business Event System or Oracle XML Gateway.

The agents must be enabled and the seeded agent listener service components must be running.

**Note:** Additional agents are provided for Web services. See Web Services Setup, page 8-4.

Use the Workflow Administrator: Agents window to enable the required agents if necessary (the agents are enabled by default). The same window can be used to disable an agent.

Use Oracle Applications Manager to schedule and manage agent listener service components. For more information, refer to the Oracle Applications Manager online help.

Ensure that the seeded agent listener service components named Workflow Deferred Agent Listener and Workflow Error Agent Listener, provided by Oracle Workflow, are running for the WF\_DEFERRED and WF\_ERROR agents, respectively.

The agent listener service components ECX Inbound Agent Listener and ECX Transaction Agent Listener are seeded with a manual startup mode and therefore must be manually started. The startup mode can be changed to "Automatic" using the Oracle Applications Manager, if desired.

Place any user-defined agents in their own container.

## Development Guidelines for Custom Messages for B2B Integration

If you are using Oracle XML Gateway to create new inbound and outbound messages, use the following guidelines to raise business events and define event subscriptions.

A prerequisite to this process is a message map created using the XML Gateway Message Designer.

See the Message Designer section, page 2-1 for instructions on how to create a new message map and how to modify an Oracle prebuilt message map.

## Development Guidelines for Outbound Messages

Following are the guidelines for developing outbound messages:

1. Register the Business Event and Corresponding Event Subscription

Register the business event and corresponding event subscription defined in the Oracle E-Business Suite.

See: Manage Workflow Processes, page 6-49 for the instructions on registering business events and event subscriptions.

2. Raise a Business Event in the Application

Use the Workflow function to raise business events in your application at points indicating when a document has been created, changed, confirmed, or deleted. Business events may be raised in PL/SQL code, by an existing Workflow process using the Raise Document Delivery Event activity, or as the first activity of a new Workflow process.

It is important to raise an event at all possible event points even though you may not define a specific event subscription for it. This will allow you to define an event subscription for any event of interest without having to modify the application code to raise the event.

3. Define an Event Subscription

Add an event subscription for each business event of interest. Use the function and event activities provided in the XML Gateway Standard Item Type to define the Workflow process associated with the event subscription.

See: Configure Oracle Prebuilt Outbound Messages, page 6-45 for the Workflow process configuration options.

See: Manage Workflow Processes, page 6-49 for the instructions on adding an event subscription.

There is no need to add an event subscription for business events that you are not interested in. These may be added at a later point in time when the event becomes of interest to your implementation.

## Development Guidelines for Inbound Messages

The guidelines for developing inbound messages are as follows:

1. Register the Business Event and Corresponding Event Subscription

Register the business event and corresponding event subscription defined in the Oracle E-Business Suite.

See: Manage Workflow Processes, page 6-49 for the instructions on registering business events and event subscriptions.

2. Set Event Details in the Message Map

Make sure your message map includes a post process Procedure Call action to call the Workflow function to set event details indicating XML Gateway has successfully processed the inbound message.

See: Procedure Call: Execute Procedure, page 2-78 for details regarding the Procedure Call action.

3. Define an Event Subscription

An event subscription is required for inbound messages from Trading Partners (B2B) regardless of whether the application is interested in the inbound message.

Assuming your application is interested in the inbound message, define an event subscription with a rule function and Workflow process relevant to the business requirement.

See: Configure Oracle Prebuilt Outbound Messages, page 6-45 for the Workflow process configuration options.

See: Manage Workflow Processes, page 6-49 for the instructions on addition a Workflow process.

There is no need to add a subscription for business events that you are not interested in. These can be added later if the business event becomes of interest to your implementation.

## Common Questions

The following are common implementation questions:

1. **Is the combination of the Generate Trading Partner XML Document and Send Document function activity valid?**

No. This combination of function activities is invalid.

The Generate Trading Partner XML Document function returns the XML message in the Event Message attribute. The Send Document function gathers the data, creates the XML message, and enqueues it onto the outbound queue.

The result of this combination of function activities is that the Send Document function activity will complete successfully but the Generate Trading Partner XML Document function activity will have data in the Event Message attribute that is not

sent anywhere unless something is defined in the Out Agent attribute (associated with the Event Subscription) which will enqueue the message.

2. **Is the combination of the Generate XML Document and Send Document function activity valid?**

Same as answer 1.

3. **Does the Generate Trading Partner XML Document function activity require the Transaction Delivery Required function activity to determine if the Trading Partner is defined to receive a document?**

Yes. The Generate Trading Partner XML Document function activity must be preceded by the Transaction Delivery Required function activity.

## Message Delivery Status

XML Message Delivery Callback allows messaging systems (such as Oracle Transport Agent) to report message delivery status back to the Workflow process that initiated message creation. If delivery fails, a Workflow notification is sent to the System Administrator who has the option to retry/resend or abort/ignore the process.

## Use of XML Message Delivery Callback

Prior to the introduction of this feature, the common Workflow process to send an outbound document to a Trading Partner was the Send Document function activity as follows:



The Send Document function activity consists of message creation and message enqueue onto the ECX\_OUTBOUND queue. If XML message delivery fails, the entire Workflow is re-executed causing the message creation step to be performed again unnecessarily.

With message delivery callback, the message creation is separated from the message enqueue, so that only the message enqueue is re-executed if the message delivery fails. The new model consists of the Generate Trading Partner XML Document function activity and the Workflow Send as follows:



The Generate Trading Partner XML Document function activity is responsible for gathering the message data. The Workflow Send event activity is responsible for enqueueing the message onto the ECX\_OUTBOUND queue.

The Oracle E-Business Suite application module delivers a Workflow with the event subscription in support of outbound documents that follows this model. Both models are valid. As with any seeded event subscription, you have the option to disable it. For information on how to disable a seeded event subscription, see: Manage Workflow Processes, page 6-49.

There are four ways that XML Message Delivery Callback can be used. They are summarized as follows:

## **Block Mode = Y**

Block Mode is an attribute of the Workflow Send event activity. A Block Mode value of "Y" implies that your Workflow process will await the message delivery status before proceeding to the next activity in the Workflow process. If delivery fails, the Workflow Default Error Process is executed to send a notification to the System Administrator. The System Administrator has the option to "Retry" the process assuming the error was corrected, or "Ignore" the process if the error cannot be corrected.

- User Event Defined

The XML message generated by the XML Gateway is enqueued onto the ECX\_OUTBOUND queue. The messaging system (such as OTA or webMethods) dequeues the message from the ECX\_OUTBOUND queue and attempts to deliver the message to the Trading Partner. Any user-defined events (defined in the Workflow Send event) will be raised and corresponding event subscriptions will be executed.

The message delivery status is returned regardless of success or failure. If the delivery fails, the Workflow Default Error Process is executed, allowing the System Administrator to retry or abort the process.

- No User Event Defined

The XML message generated by the XML Gateway is enqueued onto the ECX\_OUTBOUND queue. The messaging system (such as OTA or webMethods) dequeues the message from the ECX\_OUTBOUND queue and attempts to deliver the message to the Trading Partner. The delivery status is returned regardless of success or failure. If the delivery fails, the Workflow Default Error Process is executed, allowing the System Administrator to retry or abort the process.

## **Block Mode = N**

A Block Mode value of "N" implies that your Workflow process will proceed to the next activity on the Workflow process without waiting for the message delivery status. If delivery fails, the FYI\_MESSAGE\_DELIVERY\_ERROR process is executed to send a notification to the System Administrator.

The System Administrator has the option to "Resend" the message assuming the error was corrected or "Ignore" the notification if no action is required.

The (FYI) Message Delivery Error is a seeded process that is part of the XML Gateway Error Processing Item Type.

- User Event Defined

The XML message generated by the XML Gateway is enqueued onto the ECX\_OUTBOUND queue. The messaging system (such as OTA or webMethods) dequeues the message from the ECX\_OUTBOUND queue and attempts to deliver the

message to the Trading Partner. Any user-defined events (defined in the Workflow Send event) will be raised and corresponding event subscriptions will be executed.

If the delivery fails, the `apps.ecx.processing.message.callback` event is raised. The corresponding event subscription for the `FYI_MESSAGE_DELIVERY_ERROR` process is executed allowing the System Administrator to resend or ignore the message notification.

- **No User Event Defined**

The XML message generated by the XML Gateway is enqueued onto the `ECX_OUTBOUND` queue. The messaging system (such as OTA or webMethods) dequeues the message from the `ECX_OUTBOUND` queue and attempts to deliver the message to the Trading Partner.

If the delivery fails, the `apps.ecx.processing.message.callback` event is raised. The corresponding event subscription for the `FYI_MESSAGE_DELIVERY_ERROR` process is executed allowing the System Administrator to resend or ignore the message notification.

For more details, see *XML Gateway Error Processing Item Type*, page 6-24.

## **How Other Messaging Systems Use XML Message Delivery Callback**

Other messaging systems such as webMethods and iAS can use the XML Message Delivery Callback feature by calling the `ECX_ERRORLOG.external_system` API. The API is used by all messaging systems (both Oracle and non-Oracle) to report message delivery status. The status information is written to the XML Gateway log tables to track and report message delivery data.

For details on the API see: `ECX_ERRORLOG.external_system`, page F-2.





---

# Oracle Transport Agent

This chapter covers the following topics:

- Oracle Transport Agent Overview
- Authentication Methods
- Enabling Client Authentication
- Setup Parameters
- Connecting to Non-OTA Servers
- Code Connection Samples
- Troubleshooting

## Oracle Transport Agent Overview

Oracle Transport Agent (OTA) is a lightweight messaging platform for transmitting documents over HTTP and Secure HTTP (HTTPS). OTA implements a messaging protocol on top of the HTTP Application protocol.

The OTA server is a Java-based servlet that uses the OTA messaging protocol to support the following requirements:

- Guaranteed, exactly-once delivery of a message over HTTP(S)
- Complete audit and history tracking of messages sent and received
- Outbound e-mail delivery of messages (SMTP)
- Server certificate authentication (when using SSL mode)
- Client certificate authentication
- Built-in Application user authentication to Oracle e-Business Suite and Oracle Exchange

This section will describe how client authentication is implemented in the latest release of OTA. Oracle E-Business Suite users may choose to implement client authentication. It is not mandatory.

In this document, the Sender refers to the "Client" that sends a document or requests connection. The Receiver refers to the "Server" that receives the document or connection request.

**Note:** For information on setting up the Oracle Transport Agent, see OracleMetaLink note 152775.1, "Installing Oracle XML Gateway and Oracle Workflow with Oracle Applications 11i."

## The Oracle Transport Agent Protocol Stack

The OTA protocol stack is as follows:

**OTA (Messaging Protocol)**

**HTTP (Application Protocol)**

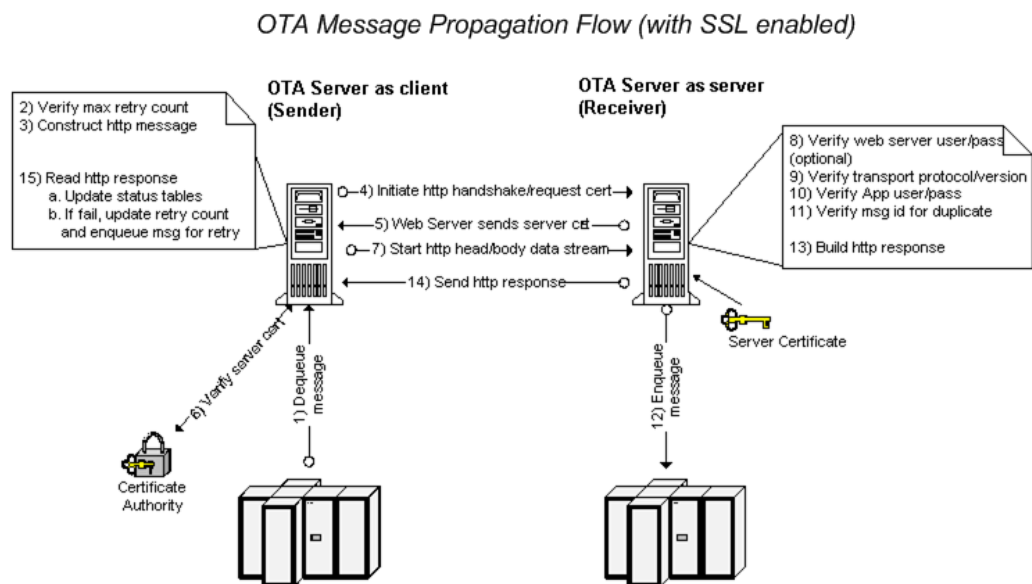
**SSL (Encryption/Security Protocol - optional)**

**TCP/IP (Network Protocol)**

The OTA protocol defines the conversation semantics used by two Web servers running the OTA Servlet. The two OTA Servlets "talk" to each other to provide guaranteed, exactly-once delivery of the message.

## OTA Message Propagation Flow

The following diagram displays the message propagation flow of an OTA message with SSL enabled:



1. The OTA client dequeues the message from the ECX\_OUTBOUND queue.
2. The OTA client verifies the maximum retry count.
3. The OTA client constructs the HTTP message.
4. The client initiates the HTTP handshake and request for certification with the destination server.
5. The destination Web server sends its certification.
6. The client receives and verifies the server's certification.

7. If verified, the client starts the HTTP message head and body data stream.
8. The server optionally verifies the username and password.
9. The server verifies the transport protocol and version.
10. The server verifies the Applications username and password.
11. The server verifies that the message ID is not a duplicate.
12. The server enqueues the message for consumption.
13. The server builds the HTTP response message.
14. The server sends the HTTP response back to the client.
15. The client reads the response and updates the status tables. If the response indicates the message failed, the retry count is updated and the message is enqueued for retry.

### **Oracle Transport Agent Post Message**

Two OTA servers communicate by sending and receiving a series of name/value pairs in the HTTP body of an HTTP POST/RESPONSE. Following is an example post from the sending OTA server (Note: the header authorization encryption follows the W3C standard):

```

HTTP Header
Http-Version: HTTP/1.1
Authorization: Digest username="myusername",
realm="testrealm@host.com",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="/dir/index.html",
qop=auth,
nc=00000001,
cnonce="0a4f113b",
response="6629fae49393a05397450978507c4ef1",
opaque="5ccc069c403ebaf9f0171e9517f40e41"
Content-length: 12345
Content-type: text/html
HTTP Body
TRANSPORT_PROTOCOL=OXTA
TRANSPORT_PROTOCOL_VERSION=1.0
REQUEST_TYPE=SEND
MESSAGE_ID=A1234567890ZZ0987654321
MESSAGE_TYPE=XML
MESSAGE_STANDARD=OAG
TRANSACTION_TYPE=PO
TRANSACTION_SUBTYPE=PROCESS
DOCUMENT_NUMBER=12345
PARTYID=9999
PARTY_SITE_ID=8888
PROTOCOL_TYPE=HTTPS-OXTA
PROTOCOL_ADDRESS=HTTPS://www.me.com/servlets/oracle.ecx.oxta. tra
nsportAgentServer
USERNAME=myusername
PASSWORD=myloginpassword
ATTRIBUTE1=
ATTRIBUTE2=
ATTRIBUTE3=
ATTRIBUTE4=
ATTRIBUTE5=
PAYLOAD=<xml ... >

```

The HTTP Body contains the message envelope, message payload, and the following transport parameters:

## TRANSPORT\_PROTOCOL

The Transport Protocol specifies the messaging protocol to the receiving servlet. The value will always be "OXTA" when sent by an OTA server. The receiving OTA server validates that the TRANSPORT\_PROTOCOL is OTA, and then sends back the appropriate response in the HTTP response body.

## TRANSPORT\_PROTOCOL\_VERSION

The Transport Protocol Version specifies the version of the messaging protocol used to send the message

## REQUEST\_TYPE

This parameter indicates the type of request sent from the OTA server. REQUEST\_TYPE is not required on the receiving side. If no value is set, the default "SEND" type is used. Valid values are in the following list:

- SEND - (Default) The receiving OTA server treats this as a live, production message.

- AUTH - (Authorization test) The receiving OTA server treats this as a request to test the communication according to the OTA AUTH protocol. This request type requires only the following parameters:

Parameter	Sample Value
TRANSPORT_PROTOCOL	OXTA
TRANSPORT_PROTOCOL_VERSION	1.0
REQUEST_TYPE	AUTH
USERNAME	MYUSERNAME
PASSWORD	MYPASSWORD

If the authorization passes, the response will contain an HTTP 200 response code and an OXTA 1000 status message. Refer to HTTP Status Codes, page 7-22 for a list of status codes.

- AUTH2 -(Authorization test, method 2) The receiving OTA server treats this as a request for a test according to the OTA AUTH2 test protocol. This request type requires only the following parameters:

Parameter	Sample Value
TRANSPORT_PROTOCOL	OXTA
TRANSPORT_PROTOCOL_VERSION	1.0
REQUEST_TYPE	AUTH2
USERNAME	MYUSERNAME
PASSWORD	MYPASSWORD
PARTY_SITE_ID	123
TRANSACTION_TYPE	PO
TRANSACTION_SUBTYPE	PROCESS

- EME - (Email me) The receiving OTA server receives the message and places it on the outbound queue for delivery to the address specified in the PROTOCOL\_ADDRESS parameter. This request type requires only the following parameters:

Parameter	Sample Value
TRANSPORT_PROTOCOL	OXTA
TRANSPORT_PROTOCOL_VERSION	1.0
REQUEST_TYPE	EME
USERNAME	MYUSERNAME
PASSWORD	MYPASSWORD
PROTOCOL_ADDRESS	me@mydomain.com
PAYLOAD	<xml....>

## MESSAGE\_ID

This is the unique message identifier set by the sending OTA server. The receiving OTA server uses this identifier to determine if the message is a duplicate. The identifier is the Oracle AQ message ID from the outbound XML Gateway queue.

## MESSAGE\_TYPE

This indicates the type of content in the payload. When used with XML Gateway, the value will always be "XML".

## TRANSACTION\_TYPE

Used by XML Gateway to determine the type of document sent. This is a user-defined value entered in the External Transaction Type field in the Transactions form. Sample values include PO, Invoice, and Shipping. For more information see Define Transactions, page 3-7.

## TRANSACTION\_SUBTYPE

Used by XML Gateway to determine the transaction subtype of the document sent. This is a user-defined value entered in the External Transaction Subtype field in the Transactions form. Sample values include Create or Change. For more information see Define Transactions, page 3-7.

## DOCUMENT\_NUMBER

This is the primary identifier of the business document in the payload. Examples include purchase order number and invoice number. This parameter is used in the Transaction Monitor as an identifier when checking the status of a document.

## PARTYID

The sender/receiver negotiated identifier that identifies the receiver of the document (company level). This is a user-defined value that is entered in the External Source Location Code of the Trading Partner form. For more information see: Define Trading Partners, page 3-13.

## PARTY\_SITE\_ID

The sender/receiver negotiated identifier that identifies the receiver of the document (company site level). The value of this parameter is the SOURCE\_TP\_LOCATION\_CODE

from the Trading Partner form. For more information see: Define Trading Partners, page 3-13.

## PROTOCOL\_TYPE

This is the application protocol to transmit the document. It also contains an identifier for the program to use to transmit the document over the protocol. The following table lists the valid values for PROTOCOL\_TYPE. For more information, see Protocol Type, page 4-4.

PROTOCOL_TYPE	Description
HTTP	Straight HTTP post
HTTP-ATCH	Straight HTTP post with attachment
HTTPS	Straight HTTP post using SSL
HTTPS-ATCH	Straight HTTP post with attachment, using SSL
HTTP-OXTA	OTA Protocol over HTTP (guaranteed delivery)
HTTPS-OXTA	OTA Protocol over HTTPS (guaranteed delivery)
OTAH-ATCH	OTA Protocol over HTTP (guaranteed delivery) with attachment
OTAH-ATCH	OTA Protocol over HTTPS (guaranteed delivery) with attachment
SMTP	Send document via SMTP (e-mail)

## PROTOCOL\_ADDRESS

This is the fully-qualified address to which to transmit the document. The following table lists sample protocol addresses for the protocol types shown:

PROTOCOL_TYPE	Sample PROTOCOL_ADDRESS
HTTP	http://www.me.com:8080/servlets/mycustom
HTTP-OXTA	http://www.me.com:9000/servlets/oracle.apps.ecx.oxta.TransportAgentServer
HTTPS-OXTA	http://www.me.com/servlets/oracle.apps.ecx.oxta.TransportAgentServer
SMTP	Me@mydomain.com

## USERNAME/PASSWORD

This is the username/password to authenticate on the receiving server. The OTA server uses this username and validates it against valid Applications users or valid Exchange users (buyer/supplier login). The username/password values are also in the authorization section of the HTTP header for optional Web server-level authentication.

#### ATTRIBUTE1

This contains the identifier of the system sending the message. This is a user-defined value entered in the ECX\_OAG\_LOGICALID profile option. See Define System Profile Options, page 3-2.

#### ATTRIBUTE2

Not used.

#### ATTRIBUTE3

This contains the identifier of the final destination for the document, when dynamic routing is implemented. This is a user-defined value entered in the Target Location Code field in the Trading Partner definition form. It is a routing mechanism to notify the receiver to route the document to another trading partner known to the receiver.

For more information about dynamic routing, see: Static and Dynamic Routing , page 3-20.

#### ATTRIBUTE4

Not used.

#### ATTRIBUTE5

Not used.

#### PAYLOAD

Contains the XML document to be processed.

### Oracle Transport Agent Response Message

The OTA server uses standard HTTP response codes to determine if the HTTP post (at the HTTP protocol level) was successful. If successful, OTA will examine the HTTP header to determine if the OTA message was successfully delivered. If the HTTP response does not equal 200, OTA assumes the post failed and will requeue the message for retry (until maximum retry has been reached).

The following is an example of an HTTP response:

```
HTTP Header
HTTP/1.1 200 OK
Server: Apache/1.2.0
Date: Fri, 15 Jun 2002 16:20:46 GMT
Content-Length: 567
STATUS_CODE: 1000
STATUS_DESCRIPTION: OK
MESSAGE_RECEIPT_ID: A9876543210987654321
Content-type: text/html
```

```
HTTP Body
<HTML><BODY> <TABLE>   <TR><TD>Status Code</TD><TD>1000</TD></TR>
<TR><TD>Status Description</TD><TD>Message Successfully received<
/ TD></TR> <TR><TD>Message Receipt ID</TD><TD>A9876543210987654321
</TD> </TR> </TABLE> </BODY></HTML>
```

The HTTP Response body is created for information only and is not read by the sending OTA server. The sending OTA server uses the STATUS\_CODE in the HTTP Response header to determine the success or failure.



If the Status Code value is 1000 (meaning success), the MESSAGE\_RECEIPT\_ID in the HTTP Response body contains a unique identifier generated by the receiving OTA server. This completes the cycle for the guaranteed delivery process.

If the Status Code is anything other than 1000, a failure has occurred. The STATUS\_DESCRIPTION field will contain a brief description of the error. Refer to HTTP Status Codes, page 7-22 for a list of status codes.

## **OTA and Attachments**

For outbound documents, the OTA client uses information in the Message Map to fetch and bundle the attachment in its outbound package.

For inbound documents, OTA extracts the associated attachments from the inbound MIME message and deposits them into the FND module of the receiving instance.

For more information about how OTA handles attachments, see Attachments and Oracle Transport Agent (OTA), page 2-91.

## **Authentication Methods**

OTA operates according to the security layer implemented at your site. There are two authentication methods supported by SSL: Server Authentication and Client Authentication.

### **Server Authentication**

When a client connects to a Web server securely via HTTPS, the server sends back its server certificate to the client for verification. Once verified, the client sends the data, encrypted, to the server. Server Authentication allows the client to identify the server.

### **Client Authentication**

The server identifies the client by requesting its client certificate. A client certificate is a digital certificate that certifies the identity of the client. A client certificate binds a system name to a specific secret key. Both the client and server certificates are issued by a Certification Authority (CA). Oracle E-Business Suite users can choose to support any CA they wish. Examples of CAs are Identris and Verisign.

### **Advantages of Client Authentication**

Client authentication supplements the traditional username/password application-level security with encrypted, tamper-proof, digital certificates for transport-level security.

Client certificates eliminate anonymity. Every connection to the server from a client requires a client's certificate and a public key that verifies the certificate.

### **Implementation of Client Authentication**

OTA enables client certificate authentication through a set of Java programs.

The SSL handshake begins when an SSL client connects to an SSL server. The SSL handshake protocol is used to authenticate the SSL server to a client and the client to the server. It also contains agreed upon encryption algorithm and keys.

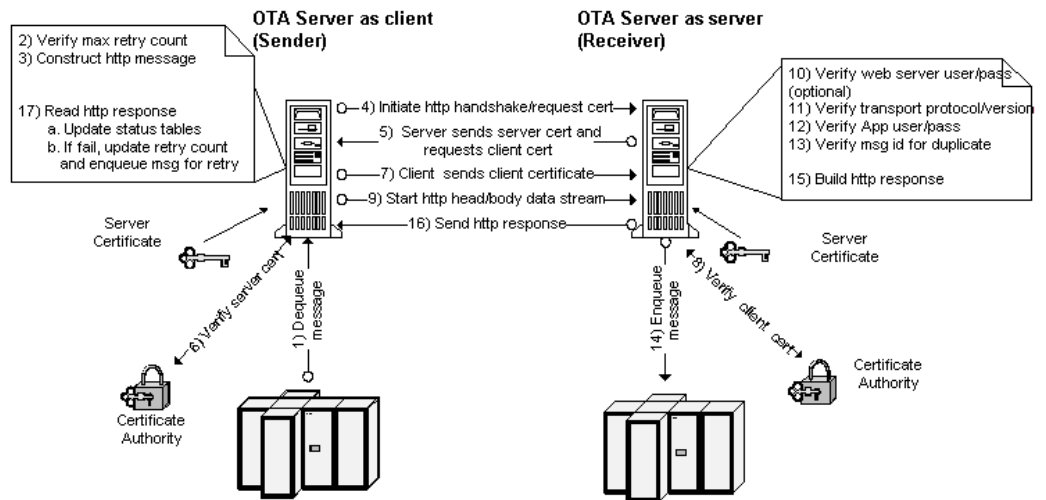
## Sequence of Events

1. The client opens a connection and sends the ClientHello. This message contains the highest SSL version understood by the client, the session ID, and the list of compression methods supported.
2. The server responds. When the SSL server receives the ClientHello, it responds with either a handshake failure alert or a ServerHello message (that is, the SSL version used by the client, the session ID, and the compression method chosen by the server for this session).
3. The server sends the certificate along with the CA (who signed the certificate) certificate chain.
4. The client verifies the server certificate with the CA certificate in the SSL Certificate Chain File (list of CAs that the Client trusts).
5. The server sends a certificate request to the client.
6. The client sends its certificate along with the CA (who signed the certificate) certificate chain. If no certificate is available, the client sends a "no certificate" alert. It is up to the SSL server to decide what to do if a no certificate alert is received. The SSL server could continue the SSL transaction with an anonymous client or could terminate the connection by sending a data handshake failure alert.
7. The client sends a ClientKeyExchange. The client may send a key exchange message depending on the particular public key algorithm.
8. The server verifies the client certificate with the CA.
9. The application data is compressed and encrypted according to the compression method.

The OTA message propagation flow with client authentication enabled is displayed in the following diagram. The difference between this flow and the previous flow is the addition of the following two steps after the verification of the server certificate and before the start of the head/body stream by the client:

- After the client verifies the server certificate, it sends a client certificate.
- The client certificate is then verified by the server. If verified, the http head/body data stream is then started by the client.

## Client Authentication



## Enabling Client Authentication

The initial release of OTA supported server authentication in which the client authenticates the server by requesting a server certificate in SSL mode. OTA is enhanced to support client authentication as well, so that it can respond to a server request for a client certificate. This is to support trading partners who wish to prevent any connection requests from anonymous clients that may harm the server.

Client authentication for OTA is enabled through AutoConfig. For instructions on enabling client authentication, see *OracleMetaLink* note 286842.1, "Enabling Client Authentication in Oracle Transport Agent (OTA) in Oracle Applications 11i."

## Setup Parameters

The following parameters in the `xmlsvcs.properties` file configure OTA properties :

The `xmlsvcs.properties` file is managed by AutoConfig, therefore any manual changes will be overwritten the next time you run AutoConfig.

For more information about AutoConfig, see *OracleMetaLink* note 165195.1, "Using AutoConfig to Manage System Configurations with Oracle Applications 11i."

## Parameters Set Through AutoConfig

These parameters set the number of database connections in your system for inbound and outbound requests. The optimum settings for these depend on the load in your system. Setting them too high could result in performance degradation. However, setting them too low could result in transactions having to wait for an excessive time before they are able to get a connection in or out. Both parameters default to 1.

**`wrapper.bin.parameters=-DOXTAInPoolSize=1`**

This parameter sets the number of database connections available for inbound requests.

**`wrapper.bin.parameters=-DOXTAOutThreads=1`**

This parameter sets the number of database connections available for outbound requests.

## Parameters Not Set Through AutoConfig

The following parameters must be manually added to the `xmlsvcs.properties` file if you wish to change the default settings. The default settings are shown with each parameter.

### Time Parameters

The following parameters control the amount of time allowed for transactions to complete over OTA:

**`wrapper.bin.parameters=-DOXTAOutBaseTimeout=10`**

The time in seconds allowed by OTA to complete the entire process flow for an outbound request. That is, the time it takes to:

1. Open a connection.
2. Send data.
3. Receive a response.

Note that this base time is adjusted for higher payloads based on the `OutLinearTimeout` factor described below.

**`wrapper.bin.parameters=-DOXTAOutLinearTimeout=500`**

This parameter sets a factor that applies to the timeout based on the size of the payload.

The timeout for a given request is calculated based on the following formula:

$\text{Timeout} = \text{OXTABaseTimeOut} + (\text{OXTAOutLinearTimeout}/1000) * \text{payload size}/1024$

where the payload size is in bytes.

For example, for a payload of 1 MB, the timeout using the default parameter settings is:

$10 \text{ sec} + [(500/1000) * 1048576/1024] = 523 \text{ seconds}$

In other words, by default, the factor 500 adds 513 seconds for every MB of data in the payload.

**`wrapper.bin.parameters=-DOXTAOutMaxTimeout=180`**

The maximum time allowed in seconds for an OTA process to complete.

**`wrapper.bin.parameters=-DOXTAOutMaxAttempts=5`**

The number of times to attempt to send an outbound request.

**`wrapper.bin.parameters=-DOXTAOutResendDelay=1800`**

The time in seconds to wait before retrying to send a transaction.

**`wrapper.bin.parameters=-DOXTAThreadSleepTime=60`**

The amount of time in seconds the master thread sleeps between monitoring the Transport Handler threads.

**`wrapper.bin.parameters=-DOXTAMaxDbConnectAttempts=20`**

The maximum number of DB failures that can be tolerated before refraining from making any more connection attempts and going to long sleep for a time given by the parameter below.

**wrapper.bin.parameters=-DOXTABackoffTime=3600**

If the Transport Handler threads have died for `OXTAMaxDbConnectAttempts` number of times due to DB failure, the master thread refrains from further attempts and goes to sleep for a time period in seconds given by this parameter.

## Payload Size

**wrapper.bin.parameters=-DOXTAInMaxContent=1000000**

The maximum payload size in bytes.

## Debug Setting

**wrapper.bin.parameters=-DOXTALogDebugMsg=false**

Set this parameter to "true" to enable standard Apache jserv logging.

## Proxy Settings

You must set the following parameters if you are using a proxy server for your outbound requests:

**wrapper.bin.parameters=-DOXTAOutUseProxy=false**

Set to "true" if you are using a proxy server.

**wrapper.bin.parameters=-DOXTAOutProxyHost=<Your proxy server name - needed if Proxy=true>**

Insert your proxy server name.

**wrapper.bin.parameters=-DOXTAOutProxyPort=<Your proxy server port - needed if Proxy=true>**

Insert your proxy server port.

## Connecting to Non-OTA Servers

The OTA server (client) includes the capability to send documents to non-OTA servlets that do not employ the OTA messaging protocol. When sending a message, the OTA server initiates an HTTP post to transmit the document. The HTTP response from the receiving Web server indicates whether the receiver was an OTA server.

If the HTTP response does not contain the OTA protocol response body, the sending OTA server assumes the message was received by a non-OTA server. In this case, the standard HTTP response code (that is, 200 meaning "success") is used to determine the success or failure of the message. If the sending OTA server receives an HTTP-200, the message delivery is assumed to be successful. If the sending OTA server receives any other response, it is assumed the delivery failed.

Successful transmissions of documents to non-OTA servers are logged as successfully delivered to a non-OTA server. Delivering to a non-OTA server has the following disadvantages:

- No guaranteed delivery

- No guaranteed once only delivery
- No message tracking available to sender
- Message ownership completely with the servlet that received the message

## Code Connection Samples

The following are code examples for remote Web servers to post a message to the Oracle E-Business Suite using either the HTTP or HTTPS protocol. Use this as a guide to create your own code.

The code to post a message to the Oracle E-Business Suite consists of three parts: the message envelope, the message payload, and the message response.

The message envelope attributes are as follows:

- MESSAGE\_TYPE
- MESSAGE\_STANDARD
- TRANSACTION\_TYPE
- TRANSACTION\_SUBTYPE
- DOCUMENT\_NUMBER
- PARTYID
- PARTY\_SITE\_ID
- PARTY\_TYPE
- PROTOCOL\_TYPE
- USERNAME
- PASSWORD

Define these as name-value pairs in your code. The USERNAME and PASSWORD should be set to something meaningful for the receiving Web server.

The PAYLOAD corresponds to your business document represented in XML format.

The receiving Web server sends the message response back to the sending Web server. The message response includes Message Receipt ID, Status Code, and Status Description. A status code of 1000 implies the message was successfully posted.

## OTA Connection over HTTP

```
import java.sql.*;
import java.net.*;
import java.io.*;
import java.util.*;
public class SendHTTP
{
    final String[] columnNames = new String[] { "MESSAGE_TYPE", "MESSAGE_STANDARD",
"TRANSACTION_TYPE", "TRANSACTION_SUBTYPE", "DOCUMENT_NUMBER", "PARTYID", "PARTY_SITE_ID", "PARTY_TYPE",
"PROTOCOL_TYPE", "PROTOCOL_ADDRESS", "USERNAME", "PASSWORD", "ATTRIBUTE1", "ATTRIBUTE2", "ATTRIBUTE3",
"ATTRIBUTE4", "ATTRIBUTE5", "PAYLOAD" };
}
```

```

        Object[] columnVals = new Object[18];
        int numCols = 0;
        String Encoding = "UTF-8";
        URL url = null;
        String username = null;
        String password = null;
        String target = null;
        String filename = null;
        String MessageID = "SEND" + System.currentTimeMillis();
        String VALUE_SEP = "=";
        String PAIR_SEP = "&";
        String NEW_LINE = "\n";
        public SendHTTP()
        {
            numCols = columnNames.length;
            columnVals = new Object[numCols];
        }
        public void createBasicMessage() throws Exception
        {
            logMessage("Creating the message");
            StringBuffer tmp = new StringBuffer("This is a test message sent from the Sample java program.");
            columnVals[0] = "XML"; //MESSAGE_TYPE
            columnVals[1] = "OAG"; //MESSAGE_STANDARD
            columnVals[2] = "ECX"; //TRANSACTION_TYPE
            columnVals[3] = "CBODO"; //TRANSACTION_SUBTYPE
            columnVals[4] = "12" ; //DOCUMENT_NUMBER
            columnVals[5] = "206"; //PARTYID
            columnVals[6] = "206"; //PARTY_SITE_ID
            columnVals[7] = "I"; //PARTY_TYPE
            columnVals[8] = "HTTP"; //PROTOCOL_TYPE
            columnVals[9] = target;
            columnVals[10] = username ; //USERNAME
            columnVals[11] = password; //PASSWORD
            columnVals[12] = null ; //ATTRIBUTE1
            columnVals[13] = null ; //ATTRIBUTE2
            columnVals[14] = null ; //ATTRIBUTE3
            columnVals[15] = null ; //ATTRIBUTE4
            columnVals[16] = null ; //ATTRIBUTE5
            columnVals[17] = getPayload();// tmp ;//Payload
        }
        public void sendMessage() throws Exception
        {
            int bytesRead;
            int total_bytes = 0;
            int bufSize = 1024;
            char[] buf = new char[bufSize];
            logMessage("Started Sending the message.");
            url = new URL((String)columnVals[9]);
            int port = url.getPort();
            if(port == -1)
                port = 80;
            URL newurl = new URL(url.getProtocol(),url.getHost(),port, url.getFile());
            logMessage("Connecting to " + newurl.getHost() + " port " + port);
            HttpURLConnection conn = (HttpURLConnection)newurl.openConnection();

```

```

        conn.setDoOutput(true);
        conn.setDoInput(true);
        OutputStreamWriter wr = new OutputStreamWriter(conn.getOutputStream());
        String httpReq = getRequestStr();
        logMessage("Sending to URL.");
        wr.write(httpReq);
        wr.flush();
        logMessage("Response Received :-");
        logMessage("Message Receipt ID = "+conn.getHeaderField("MESSAGE_RECEIPT_ID"));
        logMessage("Status Code = "+conn.getHeaderField("STATUS_CODE"));
        logMessage("Status Description = "+conn.getHeaderField("STATUS_DESCRIPTION"));
    }
    public String getRequestStr() throws Exception
    {
        StringBuffer buf = new StringBuffer();
        StringBuffer httpBody = getHttpBody();
        long bodyLen = 0;
        bodyLen = httpBody.length();
        buf.append(httpBody);
        buf.append("\n");
        return buf.toString();
    }
    public StringBuffer getHttpHeader(long contentLength) throws Exception
    {
        StringBuffer buf = new StringBuffer();
        buf.append("POST " + url.getFile() + " HTTP/1.0\r\n");
        buf.append("Host: " + url.getHost() + ":" + url.getPort() + "\n");
        String userpass = username + ":" + password;
        buf.append("Content-type: application/x-www-form-urlencoded\n");
        buf.append("Content-length: " + contentLength + "\n");
        buf.append("\r\n");
        return buf;
    }
    public StringBuffer getHttpBody() throws Exception
    {
        StringBuffer contentBuf = new StringBuffer();
        contentBuf.append(encode("TRANSPORT_PROTOCOL"));
        contentBuf.append(VALUE_SEP);
        contentBuf.append(encode("OXTA"));
        contentBuf.append(PAIR_SEP);
        contentBuf.append(encode("TRANSPORT_PROTOCOL_VERSION"));
        contentBuf.append(VALUE_SEP);
        contentBuf.append(encode("1.0"));
        contentBuf.append(PAIR_SEP);
        contentBuf.append(encode("REQUEST_TYPE"));
        contentBuf.append(VALUE_SEP);
        contentBuf.append(encode("SEND"));
        contentBuf.append(PAIR_SEP);
        contentBuf.append(encode("MESSAGE_ID"));
        contentBuf.append(VALUE_SEP);
        contentBuf.append(encode(MessageID));
    }

```



```

        contentBuf.append(PAIR_SEP);
        for (int i = 0; i < 18; i++)
        {
            if(columnVals[i] != null)
            {
                contentBuf.append(encode(columnNames[i]));
                contentBuf.append(VALUE_SEP);
                if(columnVals[i] != null)
                    contentBuf.append(encode((String)columnVals[i]));
                else
                    contentBuf.append("");
            }
            if(i != 17 )
                contentBuf.append(PAIR_SEP);
        }
        return contentBuf;
    }

    public void logMessage(String msg)
    {
        System.out.println(msg);
    }

    public String encode(String str)
    {
        String str1 = null;
        try{
            str1 = URLEncoder.encode(str,Encoding);
        }catch(Exception e)
        {
            System.out.println("Unsupported Encoding format");
            System.exit(1);
        }
        return str1;
    }

    public String getPayload()
    {
        StringBuffer tmp = null;
        try
        {
            BufferedReader in = new BufferedReader(new FileReader(fileName));
            String s = null;
            tmp = new StringBuffer();
            while((s = in.readLine()) != null)
            {
                tmp.append(s + "\n");
            }
            in.close();
        }
        catch(Exception ex)
        {
            logMessage("Exception: in reading file");
            logMessage("Sending a string : 'This is a Test String' as payload");
        }
        if(tmp == null || tmp.length() == 0)
            return "This is a Test String";
        else
            return tmp.toString();
    }

```

```

    }
    public static void main(String args[])
    {
        //USAGE Target username password;
        SendHTTP sp = new SendHTTP();
        try
        {
            if(args.length < 4)
            {
                sp.logMessage("Usage java sample <Target> <username> <password> <filename>");
                System.exit(1);
            }
            else
            {
                sp.target = args[0];
                sp.username = args[1];
                sp.password = args[2];
                sp.filename = args[3];
                sp.createBasicMessage();
                sp.sendMessage();
                sp.logMessage("Program exiting ");
            }
        }
        catch(Exception e)
        {
            sp.logMessage("Exception = "+e);
        }
    }
}

```

## OTA Connection over HTTPS

```

import java.sql.*;
import java.net.*;
import java.io.*;
import java.util.*;
import javax.net.ssl.*;
public class SendHTTPS
{
    final String[] columnNames = new String[] { "MESSAGE_TYPE", "MESSAGE_STANDARD",
        "TRANSACTION_TYPE", "TRANSACTION_SUBTYPE", "DOCUMENT_NUMBER", "PARTYID", "PARTY_SITE_ID", "PARTY_TYPE",
        "PROTOCOL_TYPE", "PROTOCOL_ADDRESS", "USERNAME", "PASSWORD", "ATTRIBUTE1", "ATTRIBUTE2", "ATTRIBUTE3",
        "ATTRIBUTE4", "ATTRIBUTE5", "PAYLOAD" };
    Object[] columnVals = new Object[18];
    int numCols = 0;
    String Encoding = "UTF-8";
    URL url = null;
    String username = null;
    String password = null;
    String target = null;
    String filename = null;
}

```

```

String MessageID = "SEND" + System.currentTimeMillis();
String VALUE_SEP = "=";
String PAIR_SEP = "&";
String NEW_LINE = "\n";
public SendHTTPS()
{
    numCols      = columnNames.length;
    columnVals    = new Object[numCols];
}
public void createBasicMessage() throws Exception
{
    logMessage("Creating the message");
    StringBuffer tmp = new StringBuffer("This is a test message sent from the Sample java program.");
    columnVals[0] = "XML"; //MESSAGE_TYPE
    columnVals[1] = "OAG"; //MESSAGE_STANDARD
    columnVals[2] = "ECX"; //TRANSACTION_TYPE
    columnVals[3] = "CBODO"; //TRANSACTION_SUBTYPE
    columnVals[4] = "12" ; //DOCUMENT_NUMBER
    columnVals[5] = "206"; //PARTYID
    columnVals[6] = "206"; //PARTY_SITE_ID
    columnVals[7] = "I"; //PARTY_TYPE
    columnVals[8] = "HTTPS"; //PROTOCOL_TYPE
    columnVals[9] = target;
    columnVals[10] = username ; //USERNAME
    columnVals[11] = password; //PASSWORD
    columnVals[12] = null ; //ATTRIBUTE1
    columnVals[13] = null ; //ATTRIBUTE2
    columnVals[14] = null ; //ATTRIBUTE3
    columnVals[15] = null ; //ATTRIBUTE4
    columnVals[16] = null ; //ATTRIBUTE5
    columnVals[17] = getPayload();// tmp ;//Payload
}
public void sendMessage() throws Exception
{
    int bytesRead;
    int total_bytes = 0;
    int bufSize      = 1024;
    char[] buf = new char[bufSize];
    String proxyHost = "www-proxy.us.oracle.com";
    String proxyPort = "80";
    Properties systemProperties = System.getProperties();
    systemProperties.setProperty("https.proxyHost",proxyHost);
    systemProperties.setProperty("https.proxyPort",proxyPort);
    logMessage("Started Sending the message.");
    url = new URL((String)columnVals[9]);
    int port = url.getPort();
    if(port == -1)
        port = 80;
    URL newurl = new URL(url.getProtocol(),url.getHost(),port, url.getFile());
    logMessage("Connecting to " + newurl.getHost() + " port " + port);
    HttpsURLConnection conn = (HttpsURLConnection)newurl.openConnection();
    conn.setDoOutput(true);
    conn.setDoInput(true);
    OutputStreamWriter wr = new OutputStreamWriter(conn.getOutputStream

```

```

Stream());
    String httpReq = getRequestStr();
    logMessage("Sending to URL.");
    wr.write(httpReq);
    wr.flush();
    logMessage("Response Received :-");
    logMessage("Message Receipt ID = "+conn.getHeaderField("MESSAGE_RECEIPT_ID"));
    logMessage("Status Code = "+conn.getHeaderField("STATUS_CODE"));
    logMessage("Status Description = "+conn.getHeaderField("STATUS_DESCRIPTION"));
}
public String getRequestStr() throws Exception
{
    StringBuffer buf = new StringBuffer();
    StringBuffer httpBody = getHttpBody();
    long bodyLen = 0;
    bodyLen = httpBody.length();
    buf.append(httpBody);
    buf.append("\n");
    return buf.toString();
}
public StringBuffer getHttpHeader(long contentLength) throws Exception
{
    StringBuffer buf = new StringBuffer();
    buf.append("POST " + url.getFile() + " HTTP/1.0\r\n");
    buf.append("Host: " + url.getHost() + ":" + url.getPort() + "\n");
    String userpass = username + ":" + password;
    buf.append("Content-type: application/x-www-form-urlencoded\n");
    buf.append("Content-length: " + contentLength + "\n");
    buf.append("\r\n");
    return buf;
}
public StringBuffer getHttpBody() throws Exception
{
    StringBuffer contentBuf = new StringBuffer();
    contentBuf.append(encode("TRANSPORT_PROTOCOL"));
    contentBuf.append(VALUE_SEP);
    contentBuf.append(encode("OXTA"));
    contentBuf.append(PAIR_SEP);
    contentBuf.append(encode("TRANSPORT_PROTOCOL_VERSION"));
    contentBuf.append(VALUE_SEP);
    contentBuf.append(encode("1.0"));
    contentBuf.append(PAIR_SEP);
    contentBuf.append(encode("REQUEST_TYPE"));
    contentBuf.append(VALUE_SEP);
    contentBuf.append(encode("SEND"));
    contentBuf.append(PAIR_SEP);
    contentBuf.append(encode("MESSAGE_ID"));
    contentBuf.append(VALUE_SEP);
    contentBuf.append(encode(MessageID));
    contentBuf.append(PAIR_SEP);
    for (int i = 0; i < 18; i++)
    {

```

```

        if(columnVals[i] != null)
        {
            contentBuf.append(encode(columnNames[i]));
            contentBuf.append(VALUE_SEP);
            if(columnVals[i] != null)
                contentBuf.append(encode((String)columnVals[i]));
            else
                contentBuf.append("");
            if(i != 17 )
                contentBuf.append(PAIR_SEP);
        }
    }
    return contentBuf;
}

public void logMessage(String msg)
{
    System.out.println(msg);
}

public String encode(String str)
{
    String str1 = null;
    try{
        str1 = URLEncoder.encode(str,Encoding);
    }catch(Exception e)
    {
        System.out.println("Unsupported Encoding format");
        System.exit(1);
    }
    return str1;
}

public String getPayload()
{
    StringBuffer tmp = null;
    try
    {
        BufferedReader in = new BufferedReader(new FileReader(fileName));
        String s = null;
        tmp = new StringBuffer();
        while((s = in.readLine()) != null)
        {
            tmp.append(s + "\n");
        }
        in.close();
    }
    catch(Exception ex)
    {
        logMessage("Exception: in reading file");
        logMessage("Sending a string : 'This is a Test String' as payload");
    }
    if(tmp == null || tmp.length() == 0)
        return "This is a Test String";
    else
        return tmp.toString();
}

public static void main(String args[])
{

```

```

//USAGE Target username password;
SendHTTPS sp = new SendHTTPS();
try
{
    if(args.length < 4)
    {
        sp.logMessage("Usage java sample <Target> <username> <password> <filename>");
        System.exit(1);
    }
}
else
{
    sp.target = args[0];
    sp.username = args[1];
    sp.password = args[2];
    sp.filename = args[3];
    sp.createBasicMessage();
    sp.sendMessage();
    sp.logMessage("Program exiting ");
}
}
catch(Exception e)
{
    sp.logMessage("Exception = "+e);
}
}
}

```

## Troubleshooting

For information on Client Authentication issues, see Common Client Authentication Implementation Issues, page G-33.

## HTTP Status Codes

The following table lists the error response codes returned in the HTTP Response by the OTA server.

Status Code	Error Code	Description
1000	OK	Request handled successfully.
2000	ECX_OXTA_DB_UNAVAIL	Database unavailable. Cannot get connection to the database.
2001	ECX_OXTA_SERVER_ERR	Unexpected server-side error. Client should retry.
3000	ECX_OXTA_BAD_REQ	Missing TRANSPORT_PROTOCOL, TRANSPORT_PROTOCOL_VERSION, or MESSAGE_TYPE.
3001	ECX_OXTA_UNKNOWN_REQ	Invalid value for MESSAGE_TYPE.
3002	ECX_OXTA_AUTH_MISSVAL	Incomplete credentials. Username or Password not available for request.

Status Code	Error Code	Description
3003	ECX_OXTA_AUTH_FAILURE	Authentication failure. Invalid user password.
3004	ECX_OXTA_PROTCL_NOTSUPP	Oracle Transport Agent protocol version not supported by server.
3005	ECX_OXTA_PAYLOAD_NULL	Message has NULL payload.
3100	ECX_OXTA_SEND_MISSVAL	Required parameters for the SEND post are missing.
3101	ECX_OXTA_LEN_MISS	Request header content length attribute not set.
3102	ECX_OXTA_LEN_TOOLARGE	Size of content larger than specified in request header content length attribute.
3200	ECX_OXTA_AUTH2_MISSVAL	Parameter missing for AUTH2 request. Verify that the party_site_id and the transaction_type are valid.
3201	ECX_OXTA_AUTH2_FAILURE	AUTH2 request failed.
3300	ECX_OXTA_EME_MISSVAL	E-mail address or payload was not passed in the EME request.
3301	ECX_OXTA_EME_INVALID_EMAIL	Incorrect e-mail address format for EME request.
3302	ECX_OXTA_EME_SMTP_NOTSET	Mail server not set up in config.
4000	ECX_OXTA_UNKNOWN_PROTCL	Invalid protocol type for SEND request.
4001	ECX_OXTA_TIMEOUT	Time out for this transport request reached.
4002	ECX_OXTA_CLIENT_ERR	Unexpected client side exception.
4003	ECX_OXTA_MAX_ATTEMPTS	Exceeded the max number of attempts for transport.
4100	ECX_OXTA_SMTP_NOTSET	Mail server not set up in config.
4101	ECX_OXTA_INVALID_EMAIL	Incorrect e-mail address format.
4102	ECX_OXTA_MAIL_ERR	Error when trying to send mail.
4103	ECX_OXTA_MAILJAR_NOT_EXISTS	SMTP not enabled. Ensure that mail.jar is present in your Java classpath.
4104	ECX_OXTA_ACTJAR_NOT_EXISTS	SMTP not enabled. Ensure that activation.jar is present in your Java classpath.
4200	ECX_OXTA_INVALID_URL	Incorrect URL format.
4201	ECX_OXTA_PROXY_FAILURE	Cannot open connection to proxy server.
4202	ECX_OXTA_CONNECT_FAILURE	Cannot connect to host:port.

Status Code	Error Code	Description
4203	ECX_OXTA_UNKNOWN_RES	Response from the server not in a format understood by the client.
4300	ECX_OXTA_INVALID_CACERT	Failed to open the certificate file/failed to read certificate from the file.
4301	ECX_OXTA_SSLHANDSHAKE_FAILURE	Failed to perform handshake when getting SSL connection.
4302	ECX_OXTA_SSLVERICHAIN_FAILURE	Error when verifying chain certificate.
5000	ECX_OXTA_PROTOCOL_MISS	Protocol value missing from inbound message.
5001	ECX_OXTA_USERNAME_MISS	Username value missing from the inbound message.
5002	ECX_OXTA_PASSWORD_MISS	Password value missing from the inbound message.



---

## Web Services

### Web Services

In Release 11*i*, any transaction supported by XML Gateway can be sent or received as a document style Web service using the Simple Object Access Protocol (SOAP).

A generic WSDL is used to inform trading partners how to communicate with the E-Business Suite. Inbound Web services must conform to this WSDL. Outbound Web services are also constructed according to this WSDL and may be exchanged with any trading partner that conforms to the E-Business Suite Web service client.

The Web services implementation in this release utilizes the SOAP version 1.1 and Web Services Description Language (WSDL) version 1.0 standards.

**Note:** These technologies are available only with Oracle9*i* Application Server Release 1 (1.0.2.2), which includes Apache SOAP.

### Web Services Components and Features

Following are the components and features implemented for Web services in this release of the Oracle E-Business Suite.

#### Web Services Description Language (WSDL)

One of the major components for publishing a Web service is the WSDL. It enables your trading partner to call your Web service without any further details. Partners can use any third party Web service tools to call your service.

Typically, schema structure is defined as part of the WSDL. In this release, the schema structure type is defined as "anyType". This allows the Oracle E-Business Suite to accept XML documents containing any XML structure. XML Gateway knows the data structure to expect based on the transaction type based on the DTD defined in the map for the message enabled for the trading partner.

The E-Business Suite publishes the WSDL to a URL for your customers to access.

#### Simple Object Access Protocol (SOAP)

The current release of Oracle E-Business Suite supports document style Web services.

#### SOAP Client (Outbound)

The SOAP client is an agent that runs on the SOAP outbound queues. It delivers messages to the specified target URLs. The SOAP client performs the following functions:

- Packages profile-related data into the SOAP envelope header

- Assembles the SOAP MIME-compliant message
- Implements guaranteed delivery, retry logic on error, and time-to-live for a transaction
- Calls back to Workflow on the status of the SOAP request
- Maps various SOAP faults to SUCCESS or FAILURE conditions
- Forms SSL request with appropriate digital certificate information
- Provides proxy support
- Logs SOAP request-related data

The SOAP header for an outbound request is defined by the trading partner's WSDL. In this release only outbound requests to another Oracle E-Business Suite implementation are supported.

### **Guaranteed Delivery**

Delivery is guaranteed for messages exchanged between trading partners that both recognize unique message IDs. The Oracle E-Business Suite leverages the AQ message ID. This ID is guaranteed to be the same across any database system in the world.

The message IDs are maintained in a repository on the receiving side. Before receiving any message, it interrogates the repository for the ID. If the message has not been received, the Web service provider receives and commits. If the message has already been received, it ignores the message and sends a successful response.

The sending side reviews the response. If it is successful, it does not retry. If it is a failure, it retries based on two parameters: Number of attempts and time-to-live for a transaction. Once both are exhausted, the message is marked as undeliverable. If implemented, a sent to the Workflow.

### **Soap Servlet (Inbound)**

The SOAP servlet receives inbound transactions. It is responsible for the following:

- Authentication
 

The system-level User ID and Password are validated against the FND\_USER table.
- Authorization
 

The trading partner information is validated against the trading partner tables.
- Message extraction
 

The message is enqueued and any attachments are placed in the attachments table.

### **Attachments**

All incoming and outgoing messages are uniquely identified by a message ID. Attachments related to the message are stored in the FND tables. For incoming Web services, the attachments are extracted by the SOAP servlet and only references are passed on as part of the message.

For outbound Web services, the SOAP client extracts the attachment from the FND attachments table and forms a MIME-compliant Body part.

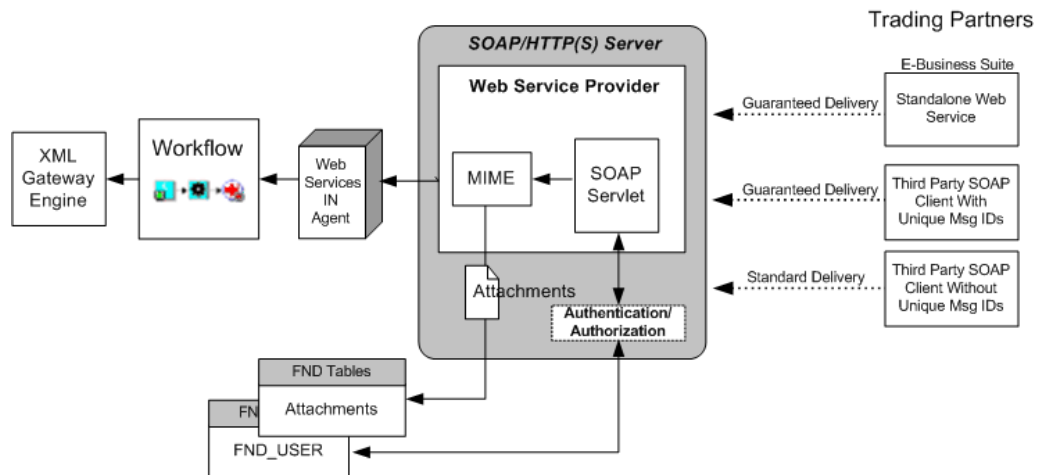
## Process Flows

This section describes the process flow of an inbound Web service and an outbound Web service in the E-Business Suite.

### Inbound Process Flow

All inbound messages are received through the SOAP servlet running under the Web service provider. Requests are authorized against the Application Object Library user tables (fnd\_user). After authorization, the body (XML data), header (trading partner information), and attachments are extracted from the request. Attachments are loaded to the FND attachment tables. XML messages are prepared as designated events before being enqueued to the SOAP agent (Web Services IN Agent). The agent listeners pick up the message for further processing. For any errors in the Web service provider, a notification is sent to the Trading Partner and/or System Administrator.

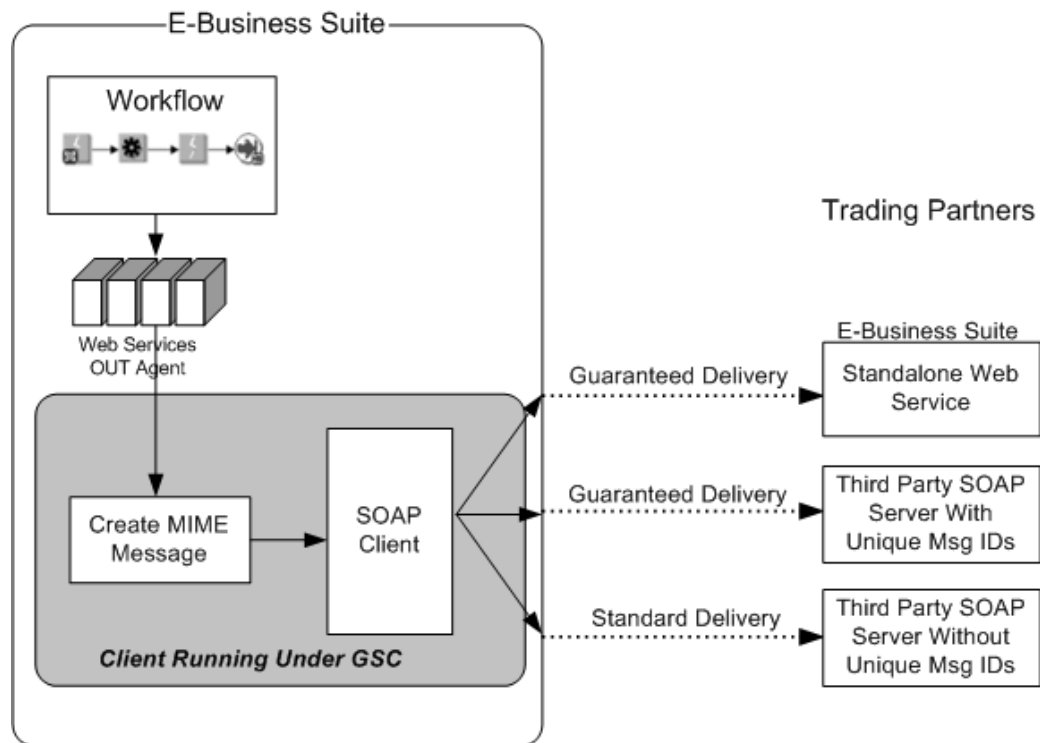
Inbound Agent Listeners monitor these agents for events. The XML message is embedded in the event, and will execute the appropriate business subscription. An event will be raised for XML Gateway.



### Outbound Process Flow

XML messages are created by Workflow processes and passed to the SOAP agent (Web Services OUT Agent). The messages are picked up by the SOAP client running under the Web service container of the Generic Service Manager (GSM). The client is responsible for the actual delivery of the message to the trading partner.

The client examines the event or message from the queue and executes the SOAP service. Messages are formatted according to the required protocol. Attachments to the message are extracted and packaged in the MIME format. References to the attachments are passed throughout the message processing. The extraction of the attachment data occurs as the final step before transmission of the transaction.



## Setup Steps

To enable Web service transactions, the following setup is required:

### Trading Partner Setup

#### Inbound

No special trading partner setup is required to enable an inbound XML message as a Document-style Web service.

For information about setting up a trading partner, see Trading Partner Setup, page 3-13.

#### Outbound

Enable your trading partner's transactions for Web services in the Trading Partner Setup form by setting the Protocol Type to SOAP and entering the Protocol Address. The Protocol Address is the URL on which the trading partner receives XML documents. For the SOAP protocol it conforms to the format: `http://<destination>/webservices/document`

To exchange Web services with a trading partner, the following information is extracted from the Trading Partner Setup form and passed as part of the SOAP header:

- Username
- Password
- Transaction Type

- Transaction Subtype
- Message Type  
Message Type must be set to "XML".
- Message Standard  
Message Type must be set to "OAG".

For more information about setting up a trading partner, see Trading Partner Setup, page 3-13.

## Set the ECX: XML Validate Flag

To send outbound Web services, the profile option ECX: XML Validate Flag must be set to "N". For more information about this profile option, see Define System Profile Options, page 3-2.

## Enable Agents and Start Agent Listeners

The following agents must be enabled and the seeded agent listener service components must be running:

Agent Name	Listener Name
WF_WS_JMS_IN	Web Services IN Agent
WF_WS_JMS_OUT	Web Services OUT Agent

Use the Workflow Administrator: Agents window to enable the agents if necessary (the agents are enabled by default). The same window can be used to disable an agent.

Use Oracle Applications Manager to schedule and manage agent listener service components. For more information, refer to the Oracle Applications Manager online help.

The agent listener components are seeded with a manual startup mode and therefore must be manually started.

## Provide the WSDL to your Trading Partner (for Inbound Transactions)

For all inbound transactions, the WSDL is published at the following URL:

`http://<IPAddress:port>/webservices/document/oracle/apps/fnd/XMLGateway?wsdl`

You can also access the WSDL by selecting **Generic XML Gateway WSDL** from the Workflow Administrator Web Applications responsibility or the Workflow Administrator Web (New) responsibility.

## Configure the Secure Socket Layer (SSL)

For inbound Web Services, SSL is handled by the Apache Server and no additional settings are required. Outbound requires additional settings. The steps required depend on what level of SSL you are enabling, Client Authentication or Server Authentication.

For all SSL communications, digital certificates will be stored in Oracle Wallet Manager. The following digital certificates are required:

- Root or Intermediate certifications of all well known Certificate Authorities
- Client Certificate
- Certificate of CA authority who signed the client certificate
- Password for accessing client certificate or the Oracle Wallet manager.

For additional configuration steps required for Web services over SSL,, see "About Oracle Workflow Mini-pack OWF.H," *OracleMetaLink* note 258312.1.

For information on setting up the Oracle Wallet Manager, see "A Guide to Understanding and Implementing SSL with Oracle Applications 11i," *OracleMetaLink* note 123718.1.

## Diagnostics

Oracle Diagnostics provides a mechanism that makes Oracle Applications more supportable and robust by providing predefined tests to check for key setup requirements. Oracle XML Gateway provides several Web Services tests through Oracle Diagnostics that you can use to check the setup and review debugging information.

You can access Oracle Diagnostics through different user interfaces, including Oracle Applications Manager and other administrative consoles. For more information, see the *Oracle Applications Supportability Guide*.

The Web Services tests are available in Oracle Diagnostics under the Application Object Library application.

### Web Services Parameter Test

This test returns the parameter settings for the Web services client. Unexpected values, such as a null value for a required parameter, will return a Warning. Review the parameter settings to ensure they contain the expected value.

### Web Service Round Trip Test

This test enqueues a sample JMS message and monitors its status. It requires the following input parameters:

- Target
- Username/Password
- Party\_Type
- Party\_ID
- PartySite\_ID
- TTL (Time to Live for the transaction)

### Web Service Class Loader Test

This test checks whether iAS 1.0.2.2 and Namespace enable parser are present in the classpath. If either is not present, the report will state that it is not present.

## Example Purchase Order Inbound Web Service

Following is an example of an inbound purchase order Web service. This example simulates a Web service from a trading partner and processes it through the Web service provider.

This example uses Axis from Apache as the Web service client. You can use any Web service client. For more information about Axis, see <http://ws.apache.org/axis/>

The example consists of three java scripts, a sample data file, and a sample payload, all of which are provided at the end of this section.

### Step 1: Generate the proxy for the Axis client.

Use the GenProxies.sh script to generate the proxy for the Axis client. The proxy is generated using the WSDL provided by the E-Business Suite.

### Step 2: Compile the code.

Use the Compile.sh script to compile the code using the generated proxies and the test data provided in the TestCase.java file. The TestCase.java file contains the message envelope and the message data.

### Step 3: Run the Web service request.

The RunTest.sh script simulates a Web service request initiated by the Axis client. RunTest requires one parameter for document number (XML element POID). The number must be unique for you to track the document via the Transaction Monitor.

### Step 4: Track the purchase order.

Once the Web service provider hands the document to the XML Gateway engine, you can monitor the document using the Transaction Monitor. Once XML Gateway has processed it, the customer order can be viewed in Order Management via the Quick Sales Order link by searching on the customer order number provided in the POID tag of the po.xml file.

## Prerequisite Files

The test scripts require the following:

### TestCase.java

TestCase.java provides the test data. It contains the message envelope and the message payload. The payload in this case is the content in the po.xml file. The envelope is coded in the the file as follows:

- header.setUsername("operations");
- header.setPassword("welcome");
- header.setPARTY\_SITE\_ID("Business World");
- header.setTRANSACTION\_TYPE("PO");
- header.setTRANSACTION\_SUBTYPE("PROCESS");
- header.setMessage\_STANDARD("OAG");
- header.setMessage\_TYPE("XML");

- `header.setDOCUMENT_NUMBER(args[0]);`

The values are for the receiving Web server. In this example, Oracle E-Business Suite is the receiving Web server, so the coded values must be valid in the XML Gateway setup for this E-Business Suite instance.

## Sample payload

The sample message is an inbound purchase order (po.xml). Note that the value in the POID tag must be unique for each test run or OrderImport will reject the transaction as a duplicate order.

## Example Scripts and Files

Following are the example scripts and files described in the introduction.

### GenProxies.sh

```
local/java/jdk1.4.2/bin/java -classpath

/wfdev/wf/wsDemo/axis-1_2beta/lib/axis.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/commons-discovery.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/commons-logging.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/jaxrpc.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/saaj.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/log4j-1.2.8.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/xml-apis.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/xercesImpl.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/wsd14j.jar

org.apache.axis.wsdl.WSDL2Java http://<IPAddress:port>/webservicess
/document/oracle/apps/fnd/XMLGateway?wsdl
```

### Compile.sh

```
/local/java/jdk1.4.2/bin/javac -classpath

./wfdev/wf/wsDemo/axis-1_2beta/lib/axis.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/commons-discovery.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/commons-logging.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/jaxrpc.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/saaj.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/log4j-1.2.8.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/xml-apis.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/xercesImpl.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/wsd14j.jar

TestCase.java
```

### RunTest.sh

```
/local/java/jdk1.4.2/bin/java -classpath

./wfdev/wf/wsDemo/axis-1_2beta/lib/axis.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/commons-discovery.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/commons-logging.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/jaxrpc.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/saaj.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/log4j-1.2.8.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/xml-apis.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/xercesImpl.jar:/wfdev/wf/wsDemo/axis-1_2beta/lib/wsd14j.jar

TestCase $1
```



## Prerequisite Files

### TestCase.java

```
import java.io.File;
import java.io.StringBufferInputStream;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.soap.SOAPException;
import org.apache.crimson.jaxp.DocumentBuilderFactoryImpl;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
;
//import com.oracleleads.le4002.oracle.apps.fnd.XMLGateway.*;
import com.oracleleads.le4010.oracle.apps.fnd.XMLGateway.*;
import com.oracle.xmlns.apps.fnd.XMLGateway.*;
import org.apache.axis.AxisFault;

public class TestCase
{
    public static void main(String[] args)
    {
        TestCase tst = new TestCase();
        if (args.length < 1 )
        {
            System.out.println("Please specify the Document Number");
            return;
        }
        System.out.println("Args :"+args[0]);
        XMLGatewaySOAPBindingStub binding = null;
        try
        {
            binding = (XMLGatewaySOAPBindingStub) new XMLGatewayLocator().getXMLGatewayPort();
        }
        catch (javax.xml.rpc.ServiceException jre)
        {
            if(jre.getLinkedCause() != null)
                jre.getLinkedCause().printStackTrace();
        }
        binding.setTimeout(60000);
        /*****Start of Populating Proxies *****/
        com.oracle.xmlns.apps.fnd.XMLGateway._ReceiveDocument_Response value
        = null;
        _XMLGateway_Header header = new _XMLGateway_Header();
        header.setUsername("operations");
        header.setPassword("welcome");
        header.setParty_Site_ID("Business World");
        header.setTransaction_Type("PO");
        header.setTransaction_Subtype("PROCESS");
        header.setMessage_Standard("OAG");
        header.setMessage_Type("XML");
        header.setDocument_Number(args[0]);
        Document body = null;
        try
        {

```

```

        DocumentBuilder builder =
            DocumentBuilderFactoryImpl.newInstance().new
DocumentBuilder();

        body = builder.parse(new File("po.xml"));
    }
    catch(Exception ex)
    {
        System.out.println("Error in loading XML:"
+ex);
    }
    /*****End of Populating Proxies *****/
    /*****End of Populating Proxies *****/
    try
    {
        value =
            binding.receiveDocument(header,body.getDoc
umentElement());

        System.out.println();
        System.out.println("Response Code
:"+value.getResponseCode());
        System.out.println("Message ID
:"+value.getResponseMsgId());
        System.out.println("Response Message
:"+value.getResponseInfo());
    }
    catch(AxisFault f)
    {
        System.out.println("Axis Fault String:
"+f.getFaultString());
    }
    catch(Exception ex)
    {
        ex.printStackTrace();
    }
}
}

```

## Sample Payload: Inbound Purchase Order

```

<!-- edited with XMLSPY v5 rel. 3 U (http://www.xmlspy.com) --> <PR
OCCESS_PO_007> <CNTROLAREA> <BSR> <VERB>PROCESS</VERB> <NOUN>PO</NO
UN> <REVISION>007</REVISION> </BSR> <SENDER> <LOGICALID/> <COMPONE
NT>IAS</COMPONENT> <TASK>POISSUE</TASK> <REFERENCEID>refid - 2583<
/REFERENCEID> <CONFIRMATION>2</CONFIRMATION> <LANGUAGE>ENG</LANGUA
GE> <CODEPAGE>US7ASCII</CODEPAGE> <AUTHID>APPS</AUTHID> </SENDER>
<DATETIME qualifier="CREATION"> <YEAR>2002</YEAR> <MONTH>10</MONTH
> <DAY>09</DAY> <HOUR>16</HOUR> <MINUTE>45</MINUTE> <SECOND>47</SE
COND> <SUBSECOND>356</SUBSECOND> <TIMEZONE>-0800</TIMEZONE> </DATE
TIME> </CNTROLAREA> <DATAAREA> <PROCESS_PO> <POORDERHDR> <DATETIME
qualifier="DOCUMENT"> <YEAR>2002</YEAR> <MONTH>10</MONTH> <DAY>09
</DAY> <HOUR>16</HOUR> <MINUTE>40</MINUTE> <SECOND>34</SECOND> <SU
BSECOND>000</SUBSECOND> <TIMEZONE>+0100</TIMEZONE> </DATETIME> <OP
ERAMT qualifier="EXTENDED" type="T"> <VALUE>107.86</VALUE> <NUMOFD
EC>6</NUMOFDEC> <SIGN>+</SIGN> <CURRENCY>USD</CURRENCY> <UOMVALUE>
1</UOMVALUE> <UOMNUMDEC>0</UOMNUMDEC> <UOM>Ea</UOM> </OPERAMT> <PO
ID>BW-562</POID> <POTYPE>Standard</POTYPE> <CONTRACTS/> <DESCRIPTN
/> <NOTES index="1"/> <USERAREA/> <PARTNER> <NAME index="1"/> <ONE

```

TIME>0</ONETIME> <PARTNRID/> <PARTNRTYPE>SoldTo</PARTNRTYPE> <PART  
 NRIDX>BWSANJOSE</PARTNRIDX> </PARTNER> </POORDERHDR> <POORDERLIN>  
 <QUANTITY qualifier="ORDERED"> <VALUE>1</VALUE> <NUMOFDEC>0</NUMOF  
 DEC> <SIGN>+</SIGN> <UOM>Ea</UOM> </QUANTITY> <OPERAMT qualifier="UN  
 IT" type="T"> <VALUE>107.86</VALUE> <NUMOFDEC>6</NUMOFDEC> <SIGN>+</SIGN> <CURRENCY>USD</CURRENCY> <UOMVALUE>1</UOMVALUE> <UOMNUMD  
 EC>0</UOMNUMDEC> <UOM>Ea</UOM> </OPERAMT> <POLINENUM>1</POLINENUM>  
 <ITEMRV/> <NOTES index="1"/> <ITEM>LAP-DLX</ITEM> <POLINESCHD> <D  
 ATETIME qualifier="NEEDELV"> <YEAR>2002</YEAR> <MONTH>10</MONTH>  
 <DAY>09</DAY> <HOUR>00</HOUR> <MINUTE>00</MINUTE> <SECOND>00</SECO  
 ND> <SUBSECOND>000</SUBSECOND> <TIMEZONE>+0100</TIMEZONE> </DATETI  
 ME> <QUANTITY qualifier="ORDERED"> <VALUE>1</VALUE> <NUMOFDEC>0</N  
 UMOFDEC> <SIGN>+</SIGN> <UOM>Ea</UOM> </QUANTITY> <PSCLINENUM>1</P  
 SCLINENUM> <USERAREA/> </POLINESCHD> </POORDERLIN> <POORDERLIN> <Q  
 UANTITY qualifier="ORDERED"> <VALUE>5</VALUE> <NUMOFDEC>0</NUMOFDE  
 C> <SIGN>+</SIGN> <UOM>Ea</UOM> </QUANTITY> <OPERAMT qualifier="UN  
 IT" type="T"> <VALUE>107.86</VALUE> <NUMOFDEC>6</NUMOFDEC> <SIGN>+</S  
 IGN> <CURRENCY>USD</CURRENCY> <UOMVALUE>1</UOMVALUE> <UOMNUMDEC  
 >0</UOMNUMDEC> <UOM>Ea</UOM> </OPERAMT> <POLINENUM>2</POLINENUM> <  
 ITEMRV/> <NOTES index="1"/> <ITEM>MON-17</ITEM> <POLINESCHD> <DATE  
 TIME qualifier="NEEDELV"> <YEAR>2002</YEAR> <MONTH>10</MONTH> <DAY>  
 Y>09</DAY> <HOUR>00</HOUR> <MINUTE>00</MINUTE> <SECOND>00</SECOND>  
 <SUBSECOND>000</SUBSECOND> <TIMEZONE>+0100</TIMEZONE> </DATETIME>  
 <QUANTITY qualifier="ORDERED"> <VALUE>5</VALUE> <NUMOFDEC>0</NUMO  
 FDEC> <SIGN>+</SIGN> <UOM>Ea</UOM> </QUANTITY> <PSCLINENUM>1</PSCL  
 INENUM> <USERAREA/> </POLINESCHD> </POORDERLIN> <POORDERLIN> <QUAN  
 TITY qualifier="ORDERED"> <VALUE>5</VALUE> <NUMOFDEC>0</NUMOFDEC>  
 <SIGN>+</SIGN> <UOM>Ea</UOM> </QUANTITY> <OPERAMT qualifier="UNIT"  
 type="T"> <VALUE>107.86</VALUE> <NUMOFDEC>6</NUMOFDEC> <SIGN>+</S  
 IGN> <CURRENCY>USD</CURRENCY> <UOMVALUE>1</UOMVALUE> <UOMNUMDEC>0<  
 /UOMNUMDEC> <UOM>Ea</UOM> </OPERAMT> <POLINENUM>3</POLINENUM> <ITE  
 MRV/> <NOTES index="1"/> <ITEM>LAP-STD</ITEM> <POLINESCHD> <DATETI  
 ME qualifier="NEEDELV"> <YEAR>2002</YEAR> <MONTH>10</MONTH> <DAY>  
 09</DAY> <HOUR>00</HOUR> <MINUTE>00</MINUTE> <SECOND>00</SECOND> <  
 SUBSECOND>000</SUBSECOND> <TIMEZONE>+0100</TIMEZONE> </DATETIME> <  
 QUANTITY qualifier="ORDERED"> <VALUE>10</VALUE> <NUMOFDEC>0</NUMOF  
 DEC> <SIGN>+</SIGN> <UOM>Ea</UOM> </QUANTITY> <PSCLINENUM>1</PSCLI  
 NENUM> <USERAREA/> </POLINESCHD> </POORDERLIN> </PROCESS\_PO> </DAT  
 AAREA> </PROCESS\_PO\_007>



---

# XML Gateway B2B Transactions Using JMS Queues

This chapter describes how XML Gateway can be integrated with any Java Messaging Service (JMS) queues so that the JMS providers can directly publish and subscribe messages to these JMS queues and leverage the Business-to-Business (B2B) transaction processing.

This chapter covers the following topics:

- Java Messaging Service (JMS) Overview
- Oracle XML Gateway and B2B Transactions Integration Points
- Oracle XML Gateway and JMS Integration
- Creating Custom JMS Queues
- Steps to Create Custom JMS Queues for Outbound B2B Transactions
- Steps to Create Custom JMS Queues for Inbound B2B Transactions

## Java Messaging Service (JMS) Overview

Java Messaging Service (JMS) is a messaging standard defined by Sun Microsystems, Oracle, and other vendors. JMS is a set of interfaces and associated semantics that define how a JMS client accesses the facilities of an enterprise-messaging product.

Oracle Java Messaging Service provides a Java API for Oracle Advanced Queuing (AQ) based on the JMS standard. Oracle JMS supports the standard JMS interfaces and has extensions to support the AQ administrative operations and other AQ features that are not a part of the standard.

## JMS Terms and Definitions

### JMS Text Message

JMS Text Message can represent a purchase order or an invoice. It contains the following components:

- Header: All messages support the same set of header properties which contain values used by both clients and providers to identify and route messages.

- **Properties:** In addition to the standard header properties, you can add optional header properties to a message. Properties can be standard properties, provider-specific, or application-specific properties.
- **Body:** This is the message payload. JMS defines various types of message payloads, as well as a type that can store JMS messages of any or all JMS-specified message types.

Oracle supports JMS Text messages using `aq$jms_text_message` or `aq$jms_message` types. Users must create the queue using one of these types as the payload.

### JMS Topic

There are sample queues created for inbound and outbound JMS messages. These two seeded queues are `WF_JMS_IN` and `WF_JMS_OUT` to receive inbound and outbound JMS messages.

**Note:** In addition to these two seeded queues, Oracle XML Gateway allows JMS providers to create customized JMS queues for Business-to-Business transactions.

For example, when a message is sent to a Trading Partner using JMS messages, Oracle XML Gateway puts the messages in the `WF_JMS_OUT` topic.

### JMS Client

A JMS client or provider is a Java-based client. A JMS client publishes a JMS Text message to a JMS topic. The Oracle Workflow Business Event System then processes this message from JMS queues.

## Oracle XML Gateway and B2B Transactions Integration Points

To provide complete support for Business-to-Business transactions, Oracle XML Gateway provides the various points of Transport protocols for Trading Partner based message consumption and message generation. These various protocol integration points are:

- Oracle Transport Agent (OTA) Using the Seeded `ECX_INBOUND` and `ECX_OUTBOUND` Queues, page 9-2
- Web Services Using Seeded `WF_WS_JMS_IN` and `WF_WS_JMS_OUT` Queues, page 9-3
- JMS Client Application Using `WF_JMS_IN`, `WF_JMS_OUT`, or Any JMS Queues, page 9-3

### Oracle Transport Agent Protocol

XML Gateway can receive and send XML messages between Trading Partners and Oracle E-Business Suite through OTA protocol using `ECX_INBOUND` and `ECX_OUTBOUND` queues.

Inbound Message Queue (`ECX_INBOUND`) holds all inbound messages that enter the process through the Transport Agent or placed directly by an API.

For outbound messages, XML Gateway creates XML messages and then enqueues them on this Outbound Message Queue (`ECX_OUTBOUND`).

See: Queues, page 5-1, Message Queues chapter for details.

## Web Services Over SOAP Protocol

XML Gateway can receive and send XML messages as a document type Web service using the Simple Object Accessed Protocol (SOAP).

All inbound messages are prepared as designed event before being enqueued to the SOAP agent, WF\_WS\_JMS\_IN queue. The Web Service IN Agent listeners pick up the message for further processing.

Outbound XML messages are created by Workflow processes and passed to the SOAP agent, WF\_WS\_JMS\_OUT queue. The messages are picked up by the SOAP client who is responsible for the actual delivery of the message to the Trading Partner.

See: Process Flows, page 8-3, Web Service chapter for inbound and outbound process flow details.

## JMS Client Application

By leveraging the Oracle Workflow Business Event System, Oracle XML Gateway allows receiving and sending JMS-based messages for business-to-business transactions between the Oracle E-Business Suite and Trading Partners using WF\_JMS\_IN, WF\_JMS\_OUT, or any JMS queues registered with Business Event System. See: Steps to Create Custom JMS Queues for Outbound B2B Transactions, page 9-5, and Steps to Create Custom JMS Queues for Inbound B2B Transactions, page 9-12.

How XML Gateway can be integrated with any JMS queues and the header properties for inbound and outbound transactions are explained in details in the next section.

# Oracle XML Gateway and JMS Integration

## Integration Features

To have seamless JMS integration and complete transaction monitoring and logging support, Oracle XML Gateway uses internal APIs to provide the following integration features:

- For inbound transactions, ability to provide complete Trading Partner user and transaction validation as well as authorization support so that XML Gateway only processes those messages that are valid and have the appropriate authorization.
- For outbound transactions, ability to provide a mechanism to verify at the runtime that the Trading Partner setup is complete and store the JMS queues as part of the setup and so that the generated messages can be sent to the desired JMS queue.

## JMS Inbound and Outbound Messages

### JMS Outbound Messages

After an application raises a business event for JMS queues, the event dispatcher catches the event, executes the event subscription to that event, enqueues the message and then places it to the default WF\_JMS\_OUT outbound agent or any customized outbound JMS queue registered with the Business Event System with the following JMS message header properties:

- ECX\_MESSAGE\_TYPE
- ECX\_MESSAGE\_STANDARD

- ECX\_TRANSACTION\_TYPE
- ECX\_TRANSACTION\_SUBTYPE
- ECX\_PARTY\_SITE\_ID
- ECX\_MSGID

A JMS provider then consumes the message placed in the JMS queue.

## JMS Inbound Messages

A JMS provider publishes XML messages to an inbound JMS agent (WF\_JMS\_IN or any JMS queue) registered with the Business Event System. The User to Trading Partner and Trading Partner to Transaction validation and authorization will then be processed so that Oracle XML Gateway can process the valid message for the inbound transaction.

The following properties must be set in the JMS message header to enable Oracle XML Gateway to process the inbound message:

- ECX\_MESSAGE\_TYPE
- ECX\_MESSAGE\_STANDARD
- ECX\_TRANSACTION\_TYPE
- ECX\_TRANSACTION\_SUBTYPE
- ECX\_PARTY\_SITE\_ID
- BES\_EVENT\_NAME=oracle.apps.ecx.jms.receive;
- BES\_EVENT\_KEY
- ECX\_USERNAME (Optional)

**Note:** The ECX\_USERNAME is used for authorization if the Enable User Security profile option is enabled. The JMS payload must contain the transaction payload.

## Impacts on the Trading Partner Setup and Seed Data

### Impacts on Trading Partner Setup

To send a message to a Trading Partner using JMS messages, the appropriate protocol type must be set first in the Trading Partner Setup form, page 3-13:

1. Select JMS as the Protocol Type field for transactions enabled at the site level.
2. Select a JMS agent registered with the Business Event System (BES) for the Protocol Address field.

The agent can be WF\_JMS\_OUT or an agent you created and registered with the BES.

## New Business Events and Event Subscriptions

### New Business Events

Oracle XML Gateway uses the following seeded event name for inbound transactions through custom JMS queue:

- Event Name: oracle.apps.ecx.jms.receive
- Event Description: Event for JMS queues



- Status: Enabled
- Owner Name: Oracle Workflow

#### Event Subscriptions for JMS Queues

Business event for JMS queues, **oracle.apps.ecx.jms.receive**, has the following seeded subscriptions:

#### Event Subscriptions for JMS Queues

Description	Phase	Status	Source Type	Rule Function	WF Process Type	WF Process Name
Rule function to perform the User to Trading Partner and Trading Partner to Transaction validation and authorization	10	Enabled	External	ECX_RULE.TPPre Processing	N/A	N/A
Rule function to perform the inbound processing	20	Enabled	External	ECX_RULE.receiveTPMessage	N/A	N/A
Rule function for error handling	10	Enabled	Error	WF_RULE.error_rule	ECXMAIN	ECXERROR

## Creating Custom JMS Queues

Oracle XML Gateway provides you with seeded JMS queues (WF\_JMS\_IN and WF\_JMS\_OUT) and a mechanism allowing you to create custom JMS queues for your B2B transaction needs.

- Steps to Create Custom JMS Queues for Outbound B2B Transactions, page 9-5
- Steps to Create Custom JMS Queues for Inbound B2B Transactions, page 9-12

## Steps to Create Custom JMS Queues for Outbound B2B Transactions

### Performing Outbound B2B Transactions Using JMS Queues:

1. Create a JMS Topic for Outbound Messages

Create a JMS Topic for outbound processing by creating queues with payload type `aq$jms_text_message`.

**Sample Code: (Create JMS topic, multiple consumers and grant the required permissions)**

Following code creates a sample JMS queue CUSTOM\_IMS\_OUT in the APPLSYS schema.

```
/* Create Queue Table for CUSTOM_IMS_OUT Topic */
declare
    queue_table_exists exception;
    pragma EXCEPTION_INIT(queue_table_exists, -24001);
begin
    dbms_output.put_line('=====');
    dbms_output.put_line('Creating Queue Table for CUSTOM_JMS_OUT T
opic ');
    dbms_output.put_line('=====');
begin
    dbms_aqadm.create_queue_table
    (
        queue_table           => 'CUSTOM_JMS_OUT',
        queue_payload_type    => 'SYS.AQ$_JMS_TEXT_MESSAGE',
        sort_list              => 'PRIORITY,ENQ_TIME',
        multiple_consumers    => TRUE,
        comment                => 'Custom JMS Topic',
        compatible             => '8.1'
    );
exception
    when queue_table_exists then
        null;
    when others then
        dbms_output.put_line('Oracle Server Error = '||to_char(sq
lcode));
        dbms_output.put_line('Oracle Server Message = '||sqlerrm
);
        raise_application_error(-20000, 'Oracle Error Mkr2= '
||to_char(sqlcode)||' - '||sqlerr
m);
end;
end;
/
commit;

/* Create Topic for CUSTOM_JMS_OUT Topic */
DECLARE
    queue_exists exception;
    pragma EXCEPTION_INIT(queue_exists, -24006);
BEGIN
    dbms_output.put_line('=====');
    dbms_output.put_line('Creating CUSTOM_JMS_OUT Topic');
    dbms_output.put_line('=====');
begin
    dbms_aqadm.create_queue
    (
        queue_name => 'CUSTOM_JMS_OUT',
        queue_table => 'CUSTOM_JMS_OUT',
        comment => 'Custom JMS Topic'
    );
exception
    when queue_exists then
        null;
    when others then
```

```

        dbms_output.put_line('Oracle Server Error = '||to_char(sqlcode));
        dbms_output.put_line('Oracle Server Message = '||sqlerrm);
;
        raise_application_error(-20000, 'Oracle Error Mkr5= '
                                ||to_char(sqlcode)||' - '||sqlerr
m);
    end;
END;
/
commit;

/* Start Topic */
declare
begin
    dbms_output.put_line('=====');
    dbms_output.put_line('Starting CUSTOM_JMS_OUT Queue ');
    dbms_output.put_line('=====');
    dbms_aqadm.start_queue(queue_name => 'CUSTOM_JMS_OUT');
    exception
    when others then
        dbms_output.put_line('Oracle Server Error = '||to_char(sqlcode));
        dbms_output.put_line('Oracle Server Message = '||sqlerrm);
;
        raise_application_error(-20000, 'Oracle Error Mkr9= '
                                ||to_char(sqlcode)||' - '||sqlerr
m);
    end;
/
commit;

/* Creates Subscribers for Multiconsumer Queues */
REM *****
*****
REM sqlplus apps/pwd@dbinst @custommoutqsubc.sql <wf schema> <wf
schema pwd>
REM *****
*****
SET VERIFY OFF
WHenever SQLERROR EXIT FAILURE ROLLBACK;
connect &1/&2
WHenever SQLERROR CONTINUE;
declare
    lagent sys.aq$_agent;
    subscriber_exist exception;
    pragma EXCEPTION_INIT(subscriber_exist, -24034);
begin
    lagent := sys.aq$_agent('CUSTOM_JMS_OUT',null,0);
    dbms_aqadm.add_subscriber(queue_name =>'&1..CUSTOM_JMS_OUT',su
bscriber=>lagent,
    rule=>'1=1');
    exception
    when subscriber_exist then
        dbms_aqadm.alter_subscriber(queue_name =>'&1..CUSTOM_JMS_OUT
',
                                subscriber=>lagent,
                                rule=>'1=1');

```

```

        null; -- ignore if we already added this subscriber.
    end;
/
commit;
exit;

/* Create the necessary grants */

REM *****
*****
REM |          APPS username
REM |          APPS password
REM |          AOL  username
REM |          AOL  password
REM *****
*****
WHenever SQLERROR EXIT FAILURE ROLLBACK;
WHenever SQLERROR CONTINUE;
WHenever OSERROR  EXIT FAILURE ROLLBACK;
    connect &3/&4
    REM create grants in APPLSYS
    grant all on custom_jms_out to &1 with grant option;
    execute dbms_aqadm.grant_queue_privilege(privilege =>'ALL', queue_name =>'&3'||'.CUSTOM_JMS_OUT', grantee =>'&1', grant_option=>TRUE);
    REM create synonyms in APPS
WHenever SQLERROR CONTINUE;
    connect &1/&2
    create synonym custom_jms_out for &3..custom_jms_out;
    commit;
exit;
/

```

## 2. Configure the Agent and Queue Handler for JMS Queues

Configure the agents and queue handler for the CUSTOM JMS topic through the Workflow Business Event Manager. Oracle Workflow provides a queue handler named `wf_event_ojmstext_qh` that provides enqueue and dequeue procedures for JMS topics.

The sample CUSTOM\_JMS\_OUT queue has the following properties:

Property	Value
Name	CUSTOM_JMS_OUT
Display Name	CUSTOM_JMS_OUT
Protocol	SQLNET
Address	<SCHEMA>.CUSTOM_JMS_OUT@<LOCAL DATABASE>
System	<LOCAL SYSTEM>
Queue Handler	WF_EVENT_OJMSTEXT_QH
Direction	OUT
Status	Enabled

3. Set Up JMS Protocol for Trading Partners in the Trading Partner Setup form, page 3-13

Use the following guidelines for custom JMS queues:

***Trading Partner Setup Guidelines for Custom JMS Queues***

Property	Value
Protocol_Type	JMS
Connection_Type	DIRECT
JMS_Queue_Name	JMS queue

4. Create a JMS Client and Subscribe to the Messages

Create a JMS Client that will subscribe to the JMS queue and consume the messages. All the message parameters will be available as part of the JMS message. The generated XML will be available as the payload in the JMS message.

**Sample Code: (JMS Subscriber)**

```
/**
 * This is a sample java file which uses Oracle JMS - Java Messag
ing Service
 * API to retrieve text Message.
 *
 * This demo does the following:
 * -- Setup Connection
 * -- Get Session from Connection
 * -- Get topic from session
 * -- Create a Durable Subscriber for this topic
 * -- Receive the Message for subscriber
 *
 * The following instructions describe how to compile and execute
 * this sample on the client machine.
 *
 * System requirements:
 * =====
 * 1) The client machine should have JDK 1.1.x or JDK1.2 or high
er installed
 * 2) The following jar/zip files should be in the CLASSPATH on
the client
 *     machine.
 *     For JDK1.2.x
 *         classes12.zip
 *         aqapi.jar
 *         jmscommon.jar
 *     For JDK1.1.x
 *         classes111.zip
 *         aqapi11.jar
 *         jmscommon.jar
 * Set up CLASSPATH, PATH, LD_LIBRARY_PATH based on JDK version a
nd platform.
 * Compilation and Running:
 * =====
 * 3) If you already have the jars in step 2) in classpath
 *     javac JMSSubscriber.java
```

```

* 4) java JMSSubscriber
***/
import javax.jms.JMSEException;
import javax.jms.Session;
import javax.jms.TextMessage;
import javax.jms.Topic;
import javax.jms.TopicSession;
import javax.jms.TopicSubscriber;
import oracle.jms.AQjmsConnection;
import oracle.jms.AQjmsFactory;
import oracle.jms.AQjmsSession;
import oracle.jms.AQjmsTopicConnectionFactory;
import oracle.jms.AQjmsTextMessage;
public class JMSSubscriber {
    /*
        * TopicConnectionFactory object used to create topic connectio
ns
        */
        AQjmsTopicConnectionFactory topicConnectionFactory = null;
        Topic topic = null;
        /* JMS Connection object */
        AQjmsConnection topicConnection = null;
        /*
            * TopicSession object for Subscribers
            */
        TopicSession subTopicSession = null;
        /**
            * Default Constructor
            */
        public JMSSubscriber() {
            try {
                /* Get Connection Factory for topic with specified TNS and
driver */
                topicConnectionFactory = (AQjmsTopicConnectionFactory) AQjms
sFactory
                                .getTopicConnectionFactory("ap601
sdb",
                                "atgwfddev", 4135, "thin");

                /* Create topicConnection with username and password */
                topicConnection = (AQjmsConnection) topicConnectionFactory
                                .createTopicConnection("apps", "apps");

                /* Create subscriber TopicSession */
                subTopicSession = topicConnection.createTopicSession(false,
                                Session.AUTO_ACKNOWLEDGE);
                /* Get topic object from schema and queue */
                topic = ((AQjmsSession) subTopicSession).getTopic("applsys"
,
                                "custom_jms_in");
            }
            /*
                * Start Connection. It is in Stopped mode by default
                */
            topicConnection.start();
        }
        catch (JMSEException ex) {
            System.out.println("Exception = " + ex.toString());
        }
    }
}

```

```

    }
    /**
     * Method to close topicConnection
     *
     * @throws JMSEException
     */
    public void close() throws JMSEException {
        topicConnection.close();
    }
    /**
     * Method to recieve messages from topic
     */
    public void recieveMessages() {
        try {
            TopicSubscriber topicSubscriber = subTopicSession
                .createDurableSubscriber(topic, "
Subscriber1");
            boolean done = false;
            System.out.println("Dequeue Messages for Subscriber ");
            AQjmsTextMessage msg = null;
            while (!done) {
                msg = (AQjmsTextMessage) topicSubscriber.receiveNoWait();
;
                if (msg == null) {
                    done = true;
                }
                else {
                    System.out.println("Recieved Message with ID : " +
                        msg.getJMSMessageID());
                    System.out.println("Message Properties -");
                    System.out.println("ECX_MESSAGE_TYPE : " +
                        msg.getStringProperty("ECX_MESSAGE_
TYPE"));
                    System.out.println("ECX_MESSAGE_STANDARD : " +
                        msg.getStringProperty("ECX_MESSAGE_
STANDARD"));
                    System.out.println("ECX_TRANSACTION_TYPE : " +
                        msg.getStringProperty("ECX_TRANSACT
ION_TYPE"));
                    System.out.println("ECX_TRANSACTION_SUBTYPE : " +
                        msg.getStringProperty("ECX_TRANSACT
ION_SUBTYPE"));
                    System.out.println("ECX_PARTY_SITE_ID : " +
                        msg.getStringProperty("ECX_PARTY_SI
TE_ID"));
                    System.out.println("BES_EVENT_NAME : " +
                        msg.getStringProperty("BES_EVENT_NA
ME"));
                    System.out.println("BES_EVENT_KEY : " +
                        msg.getStringProperty("BES_EVENT_KE
Y"));
                    System.out.println("Message Text -");
                    System.out.println(msg.getText());
                }
            }
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

    }
}
public static void main(String[] args) {
    JMSSubscriber jMSSubscriber = new JMSSubscriber();
    jMSSubscriber.receiveMessages();
    try {
        /* Close topicConnection */
        jMSSubscriber.close();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

## Related Topics

Steps to Create Custom JMS Queues for Inbound B2B Transactions, page 9-12

Java Messaging Service (JMS) Overview, page 9-1

Oracle XML Gateway and B2B Transactions Integration Points, page 9-2

## Steps to Create Custom JMS Queues for Inbound B2B Transactions

### Performing Inbound B2B Transactions Using JMS Queues:

1. Create a JMS Topic for Inbound Messages

Create a JMS Topic for inbound processing by creating queues with payload type `aq$jms_text_message`.

See: Step 1, Performing Outbound B2B Transactions Using JMS Queues, page 9-5 for details. Replace the `CUSTOM_JMS_OUT` queue with `CUSTOM_JMS_IN` to create a JMS queue for inbound messages.

2. Configure the Agent and Queue Handler for JMS Queues

See: Step 2, Performing Outbound B2B Transactions Using JMS Queues, page 9-8 for details. Replace the `CUSTOM_JMS_OUT` queue with `CUSTOM_JMS_IN` and use direction as `IN`.

3. Create a JMS Client and Publish Messages to the Custom JMS Queue

Create a JMS Client that will publish messages to the JMS queue. When publishing the messages, there are some XML Gateway specific parameters that should be available as part of the JMS message:



### **Parameters Required in the Custom JMS Message**

Name	Value
ECX_TRANSACTION_TYPE	External Transaction Type
ECX_TRANSACTION_SUBTYPE	External Transaction Subtype
ECX_MESSAGE_STANDARD	Message Standard
ECX_MESSAGE_Type	Message Type
ECX_PARTY_SITE_ID	Source TP Location Code
BES_EVENT_NAME	oracle.apps.ecx.jms.receive
BES_EVENT_KEY	Unique key for every run
ECX_USERNAME (Optional)	Valid Oracle Application Users

**Note:** The XML Payload should be set as the payload in the JMS message. Any other optional parameters specific to the processing should also be provided as part of these name value pairs.

### **Sample Code: (JMS Publisher)**

```
/**
 * This is a sample java file which uses Oracle JMS - Java Messag
ing Service
 * API to publish text Message.
 *
 * This does the following:
 * -- Setup Connection
 * -- Get Session from Connection
 * -- Setup topic
 * -- Create a publisher for this topic
 * -- Publish Message to the Topic
 *
 * The following instructions describe how to compile and execute
 * this sample on the client machine.
 *
 * System requirements:
 * =====
 * 1) The client machine should have JDK 1.1.x or JDK1.2 or high
er installed
 * 2) The following jar/zip files should be in the CLASSPATH on
the client
 *     machine.
 *     For JDK1.2.x
 *         classes12.zip
 *         aqapi.jar
 *         jmscommon.jar
 *     For JDK1.1.x
 *         classes111.zip
 *         aqapi11.jar
 *         jmscommon.jar
 * Set up CLASSPATH, PATH, LD_LIBRARY_PATH based on JDK version
and platform.
 * Compilation and Running:
```

```

* =====
* 3) If you already have the jars in step 2) in classpath
*     javac JMSPublisher.java
*
* 4) java JMSPublisher
*
*** /
import javax.jms.JMSEException;
import javax.jms.Session;
import javax.jms.TextMessage;
import javax.jms.Topic;
import javax.jms.TopicPublisher;
import javax.jms.TopicSession;
import oracle.jms.AQjmsConnection;
import oracle.jms.AQjmsFactory;
import oracle.jms.AQjmsSession;
import oracle.jms.AQjmsTextMessage;
import oracle.jms.AQjmsTopicConnectionFactory;

public class JMSPublisher {
    /*
     * TopicConnectionFactory object used to create topic connectio
ns
    */
    AQjmsTopicConnectionFactory topicConnectionFactory = null;
    Topic topic = null;
    /* JMS Connection object */
    AQjmsConnection topicConnection = null;
    /*
     * TopicSession object for Publishers.
    */
    static AQjmsSession pubTopicSession = null;
    TopicPublisher topicPublisher = null;
    /**
     * Default Constructor
    */
    public JMSPublisher() {
        try {
            /* Get Connection Factory for topic with specified TNS and
driver*/
            topicConnectionFactory = (AQjmsTopicConnectionFactory) AQjmsFactory
                .getTopicConnectionFactory("ap601sdb",
                    "atgwfdev", 4135, "thin");
            /* Create topicConnection with username and password*/
            topicConnection = (AQjmsConnection) topicConnectionFactory
                .createTopicConnection("apps", "apps");
            /* Create publisher TopicSession*/
            pubTopicSession = (AQjmsSession) topicConnection
                .createTopicSession(false, Session.AUTO_A
CKNOWLEDGE);
            /* Get topic object from schema and queue*/
            topic = ((AQjmsSession) pubTopicSession).getTopic("applsyst",
                "custom_jms_in");
            /* Create Publisher*/
            topicPublisher = pubTopicSession.createPublisher(topic);

```

```

        /* Start Connection.
        * It is in Stopped mode by default
        */
        topicConnection.start();
    }
    catch (JMSEException ex) {
        System.out.println("Exception = " + ex.toString());
    }
}

/**
 * Method to close topicConnection
 * @throws JMSEException
 */
public void close() throws JMSEException {
    topicConnection.close();
}

/**
 * Method to publish message to topic
 * @return msgid MessageId of the published Message
 */
public String publishMessage() {
    String msgid = null;
    AQjmsTextMessage msg = null;
    try {
        /*
        * Create Text Message
        */
        msg = (AQjmsTextMessage) pubTopicSession.createTextMessage();
    ;

        /*
        * Set its properties
        */
        msg.setStringProperty("ECX_MESSAGE_TYPE", "XML");
        msg.setStringProperty("ECX_MESSAGE_STANDARD", "OAG");
        msg.setStringProperty("ECX_TRANSACTION_TYPE", "INVOICE");
        msg.setStringProperty("ECX_TRANSACTION_SUBTYPE", "PROCESS");
        msg.setStringProperty("ECX_PARTY_SITE_ID", "Business World")
    ;

        msg.setStringProperty("BES_EVENT_NAME", "oracle.apps.ecx.jms.
receive");
        msg.setStringProperty("BES_EVENT_KEY", "n6");
        /*
        * Set Payload.Here its XML text
        */
        String payload = new String(
            "<ECX_MAPS><ECX_MAPPINGS><MAP_ID>1</MAP_ID><MAP_CODE>hehe<
/MAP_CODE></ECX_MAPPINGS></ECX_MAPS>");
        msg.setText(payload);
        /*
        * Publish Message to topic
        */
        topicPublisher.publish(topic, msg);

        /* retrieve message id of posted message*/
        msgid = msg.getJMSMessageID();
    }
    catch (JMSEException e) {

```

```

        e.printStackTrace();
    }

    return msgid;
}

public static void main(String[] args) {
    JMSPublisher jMSPublisher = new JMSPublisher();
    /* Post a message and get message id of posted message*/
    String msgid = jMSPublisher.publishMessage();
    /* Message Id is of Format - ID:<id>
    * Example ID:ASFG43543JH5K435H5K4
    * So retrieve correct id
    */
    msgid = msgid.substring(3, msgid.length() - 1);

    System.out.println(" Message ID = " + msgid);

    try {
        /* Close topicConnection */
        jMSPublisher.close();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

## Related Topics

[Steps to Create Custom JMS Queues for Outbound B2B Transactions, page 9-5](#)

[Java Messaging Service \(JMS\) Overview, page 9-1](#)

[Oracle XML Gateway and B2B Transactions Integration Points, page 9-2](#)

# Map Analysis Guidelines

## Map Analysis Overview

A map analysis is required to determine how to map data between the Oracle E-Business Suite data model and the required message. For outbound messages, this requires a comparison between the Oracle E-Business Suite database views and the Document Type Definition (DTD) or a production XML message.

The result of the map analysis is used as input to the XML Gateway Message Designer. It is imperative that a complete map analysis is done before attempting to create the message map. The map analysis must resolve all data gaps, identify derivation rules, identify default values, identify data transformation and process controls via Actions, and ensure that all required target fields are mapped.

- Map Analysis Guidelines for Outbound Messages, page A-1
- Map Analysis Guidelines for Inbound Messages, page A-7

## Map Analysis Guidelines for Outbound Messages

The steps to performing a map analysis for an outbound message are as follows:

- Compare database views (source) to DTD (target)
- Identify application business event trigger
- Add XML Gateway required Actions to define selection criteria
- Identify source and target document levels

## Map Analysis Guidelines for Outbound Messages Checklist

Use the following table as a checklist to track your progress when analyzing an outbound message. Details of each step are provided in the following sections. Use the "Completed" column to check off the step as it is completed.

Step	Completed	Description
1		<b>CREATE WORKSHEET:</b> Create a worksheet to identify all data.
2		<b>SET UP SOURCE:</b> Create database views or get database views/table definitions (source) for the required message from the database schema.
3		<b>SET UP TARGET:</b> Populate the target column of the worksheet with the DTD or production XML message in the expanded format.

Step	Completed	Description
3A		Identify the required DTD (target) elements.
3B		For DTD elements with "1" as the occurrence indicator, select one element from the list. Make a note of the elements not selected because these must be removed from the message map to prevent a parser violation.
4		<b>RELATE SOURCE ELEMENT TO TARGET ELEMENT:</b> For each DTD (target) element, identify the database view (source) column to map from.
4A		Identify where code conversion is needed.
4B		Identify Derivation Rules
4C		Identify Defaulting Rules
4D		Identify DTD Attribute Settings
5		<b>CONVERT FROM ORACLE FORMAT TO OAG FORMAT:</b> For each OAG DTD (target) DATETIME, AMOUNT, OPERAMT, or QUANTITY element, identify the database view/table (source) column to map from. Other XML standards may have similar elements.
5A		Identify and map each DATETIME element.
5B		Identify and map each AMOUNT element.
5C		Identify and map each OPERAMT element.
5D		Identify and map each QUANTITY element.
6		<b>ADD SIBLING FOR DUPLICATE NODES/ELEMENTS:</b> Add Sibling nodes and elements if needed.
7		<b>ADD CHILD TO EXTEND DTD DEFINITION FOR FLEXFIELDS:</b> If there is a database view or table column that you wish to map from but no DTD element is defined to support it, you can use the OAG's USERAREA. Map data to the USERAREA (or corresponding area if using other XML standards).
8		<b>RESOLVE DATA GAPS:</b> Any DTD (target) elements that have not been mapped are data gaps.
9		<b>IDENTIFY APPLICATION BUSINESS EVENT TRIGGER:</b> Identify the application business events to trigger message creation.
10		<b>IDENTIFY REQUIRED ACTIONS:</b> All outbound messages require the following Actions to define the selection criteria. Make a notation on the worksheet to include them. Use the Append Where Clause action to bind to transaction type, transaction subtype, party ID, and party site ID to ECX_OAG_CONTROLAREA_TP_V (formerly ECX_OAG_CONTROLAREA_V see Note, page 2-25). If using ECX_OAG_CONTROLAREA_V, use the Append Where Clause action to bind to transaction type and transaction subtype. Use the Append Where Clause action to bind document ID to header view to select the document
11		<b>IDENTIFY SOURCE AND TARGET DOCUMENT LEVELS</b>

## Compare Database Views (Source) to DTD (Target)

### Step 1 Create a Worksheet

Create a worksheet with the following columns: Target, Required (Y/N), Source, Code Category, Derivation Rule, Default Value, and Action, as shown in the following table:

Target	Required (Y/N)	Source	Code Category	Derivation Rule	Default Value	Action

### Step 2 Set Up Source

Create database views or get database view and table definitions (source) for the required message from the database schema. Include ECX\_OAG\_CONTROLAREA\_V or ECX\_OAG\_CONTROLAREA\_TP\_V (see Note, page 2-25) view to map to the CNTROLAREA segment of the DTD.

### Step 3 Set Up Target

Populate the target column of the worksheet with the DTD or production XML message in the expanded format.

**Note:** You may use third party software to expand the DTD or use a production XML message.

### Step 3A Identify Required Target Elements

Identify the required DTD (target) elements. An element is required if there is no special symbol next to the element name.

The XML occurrence indicators are:

"+" implies one or more

"\*" implies zero or more

"?" implies zero or 1

"|" implies either one or the other

"," implies all listed child elements must be used in the order shown

### Step 3B For Elements with "|" as the occurrence indicator

For DTD elements with "|" as the occurrence indicator, select one element from the list. Make a note of the elements not selected in order to remove them from the message map. They must be removed from the message map to prevent a parser violation.

### Step 4 Relate Source Element to Target Element

For each DTD (target) element, identify the database view (source) column to map from.

**Note:** Use the *Open Application Group's Integration Specification* (available at <http://www.openapplications.org>) Appendix C, Field Identifier Descriptions and Appendix D, Segment Descriptions, to get detailed descriptions of each element identified on the DTD.

## Step 4A

If the DTD (target) element can be sourced from a database view (source) column, determine if the database view column value requires code conversion.

- If code conversion is required, identify the code category from the list in Appendix B, page B-1.
- Identify the from or to value to determine if the required values are available in the database. If the values are not available, they must be added.

## Step 4B Identify Derivation Rule

If the DTD (target) element cannot be sourced from a database view (source) column, determine if the value can be derived. If yes, identify the derivation rule. The derivation rule can be, for example, a combination of several view columns, a procedure call, or a function call. Make a notation that an XML Gateway Action may be required (for example, to get a value from another column, or to combine the values of several columns).

Refer to Appendix C, page C-1. for a list of XML Gateway supported Actions.

Determine if the derived value requires code conversion. If code conversion is required, identify the code category from the list in Appendix B, page B-1.

**Important:** Identify the from or to value to determine if the required values are available in the database. If the values are not available, they must be added.

## Step 4C Identify Defaulting Rule

If the DTD (target) element cannot be sourced from a database view (source) column or derived, determine if the value can be defaulted. If yes, identify the default value. The default value can be a literal or based on a value from another database view column. Make a notation that an XML Gateway Action may be required.

The default value can be set directly in the map using the Message Designer or as an XML Gateway Action if the default value is based on a condition.

Refer to Transaction Map - Actions, page 2-51 for a list of XML Gateway supported Actions.

Determine if the default value requires code conversion. If code conversion is required, identify the code category from the list in Appendix B, page B-1.

**Important:** Identify the from or to value to determine if the required values are available in the database. If the values are not available, they must be added.

## Step 4D Identify DTD Attribute Setting

For each DTD (target) element containing attributes (lower case tags), determine the appropriate setting by checking the usage of the database view or table column and then identifying a valid setting for the attribute by reviewing the options in OAG's Appendix D that lists Segment Descriptions for the DTD element. The appropriate attribute setting is set directly in the target definition as a default value using the Message Designer.

For example, if a view column is used for creation date, the DATETIME "qualifier" attribute value of "CREATION" will be set as the default value using the Message Designer.



## Step 5 Convert from Oracle Format to OAG Format

For each DTD (target) DATETIME, AMOUNT, OPERAMT, or QUANTITY element, identify the database view or table (source) column to map from.

Oracle E-Business Suite represents date, amount, operating amount, and quantity in a single database column. OAG represents these as a collection of data. Oracle XML Gateway provides an Action to convert the Oracle representation to the OAG representation. Make a notation that this column requires an Action for "Convert to OAG" that will be defined using the Message Designer.

### Step 5A

- For the DATETIME element, identify the database view or table (source) column for the date.
- Follow Step 4D to set the "qualifier," "type," and "index" attributes
- All other OAG fields are set by the "Convert to OAG" Action based on the date value.
- TIMEZONE is defaulted to Greenwich Mean Time (GMT).

**Note:** OAG DATETIME is made up of the following attributes and elements: qualifier, type, index, YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, SUBSECOND, TIMEZONE.

### Step 5B

- For the AMOUNT element, identify database view or table (source) columns for currency code and debit/credit flag if available. The XML Gateway "Convert to OAG" Action can be defined to use the view or table column values or default to OAG recommended values if the view or table columns are not available.
- Follow Step 4D to set the "qualifier," "type," and "index" attributes.
- The OAG fields for NUMOFDEC and SIGN are set by the "Convert to OAG" Action based on the amount value.

**Note:** OAG AMOUNT is made up of the following attributes and elements: qualifier, type, index, VALUE, NUMOFDEC, SIGN, CURRENCY, DRCD.

### Step 5C

- For the OPERAMT fields, identify database view or table (source) columns for currency and unit of measure code if available. The XML Gateway "Convert to OAG" Action can be defined to use the view or table column values or default to OAG recommended values if the view or table columns are not available.
- Follow Step 4D to set the "qualifier" and "type" attributes.
- The OAG fields for NUMOFDEC, SIGN, UOMVALUE, and UOMNUMDEC are set by the "Convert to OAG" Action based on the operating amount value.

**Note:** OAG OPERAMT is made up of the following attributes and elements: qualifier, type, VALUE, NUMOFDEC, SIGN, CURRENCY, UOMVALUE, UOMNUMDEC, UOM.

## Step 5D

For the QUANTITY element, identify database view or table (source) column for the unit of measure code if available. The XML Gateway "Convert to OAG" Action can be defined to use the view or table column value or default to the OAG recommended value if the view or table column is not available.

Follow Step 4D to set the "qualifier" attribute.

The OAG fields for NUMOFDEC and SIGN are set by the "Convert to OAG" Action based on the quantity value.

**Note:** OAG QUANTITY is made up of the following attributes and elements: qualifier, VALUE, NUMOFDEC, SIGN, UOM.

## Step 6 Add Sibling for Duplicate Nodes/Elements

A DTD defines a single occurrence of a node or element. If you have multiple occurrences from the database view or table (source) to map from, make a notation on your worksheet showing the duplicate node or element and then map the element and attributes of the duplicate node or element. Make a notation to ADD SIBLING using the Message Designer.

An example is the DTD element for PARTNER used to map SHIP-TO, BILL-TO, and REMIT-TO from the Oracle E-Business Suite. You will need to add two PARTNER sibling nodes (same hierarchy level) to map BILL-TO and REMIT-TO.

## Step 7 Add Child to Extend DTD Definition for Flexfields

If there is a database view or table column you wish to map from but no DTD element is defined to support it, you can use the USERAREA. You can add a USERAREA as a sibling (same hierarchy level) of another USERAREA or extend an existing USERAREA by adding a child (next level of hierarchy) element and then mapping the element. Make a notation to ADD CHILD using the Message Designer.

This approach is necessary to support all the flexfields included in the database views and tables. Since we do not know which flexfields are implemented at a user's site, all flexfields must be mapped.

Refer to How to Extend DTDs, page 2-84 for details.

**Note:** The only updates allowed for a DTD are to the USERAREA. Any other changes will invalidate the DTD.

## Step 8 Resolve Data Gaps

At this point, you have identified database view and table (source) columns for as many DTD (target) elements as possible. Any required DTD (target) element that has not been mapped represents a data gap.

Consider resolving the data gap using an application flexfield that is included in the database view or table. If this is a valid option, you must implement the designated flexfield by defining and populating it in the Oracle E-Business Suite. Consider carefully which flexfield to implement as you do not want to overwrite a flexfield implemented by the user.

Consider resolving the data gap using an application column not included in the database view. If this is a valid option, you must modify the database view to include the

application column as opposed to referencing the table and column directly. Load the modified database view into the database schema. Remember to map the field.

If neither of the above considerations resolves the data gap, then you must add functionality to your application module, create new database views, load the new views into the database schema, and then create the message map.

### Step 9 Identify Application Event Trigger

Identify the application business events to trigger message creation. The common trigger points are when the document is created, confirmed, updated, or deleted.

Use the Oracle Workflow Business Event System to register the business event to define the corresponding event subscription to create and send the message.

See: Integrating Oracle XML Gateway with Oracle Workflow Business Event System, page 6-1 for details on how to register and subscribe to application business events.

### Step 10 Identify Required Actions

All outbound messages require the following Actions to identify the selection criteria. Make a notation on your worksheet to include them.

- If you are using ECX\_OAG\_CONTROLAREA\_TP\_V use the Append Where Clause action to bind transaction type, transaction subtype, party ID, and party site ID

If you are using ECX\_OAG\_CONTROLAREA\_V use the Append Where Clause action to bind transaction type and transaction subtype

**Note:** The ECX\_OAG\_CONTROLAREA\_TP\_V view is an upgraded version of the ECX\_OAG\_CONTROLAREA\_V view. Oracle XML Gateway supports both versions of the database view. The more information see the Note, page 2-25.

- Use the Append Where Clause action to bind document ID to the header view to select the document

### Step 11 Identify Source and Target Document Levels

See: Identifying Source and Target Document Levels, page A-13.

## Map Analysis Guidelines for Inbound Messages

The map analysis process for inbound messages consists of the following steps:

- Compare DTD (source) to Application Open Interface tables (target)
- Add XML Gateway required Actions
- Add XML Gateway optional Actions
- Identify source and target document levels

### Map Analysis Guidelines for Inbound Messages Checklist

Use the following table as a checklist to track your progress when analyzing an inbound message. Details of each step are provided in the following sections. Use the "Completed" column to enter completion data or to check off the step when completed.

Step	Completed	Description
1		<b>CREATE WORKSHEET:</b> Create a worksheet to identify all data.
2		<b>SET UP SOURCE:</b> Get the DTD (source) for the required message or use a production XML message.
3		<b>SET UP TARGET:</b> Populate the target column of the worksheet with the Application Open Interface table definitions or Application API parameter list.
3A		Identify the required Application Open Interface (target) table columns or Application API parameters.
4		<b>RELATE SOURCE ELEMENT TO TARGET ELEMENT:</b> For each Application Open Interface (target) table column, identify the DTD (source) element to map from.
4A		Identify where code conversion is needed.
4B		Identify Derivation Rules
4C		Identify Defaulting Rules
5		<b>CONVERT FROM OAG FORMAT TO ORACLE FORMAT:</b> For each OAG DTD (source) DATETIME, AMOUNT, OPERAMT, or QUANTITY element, identify the Application Open Interface (target) column or Application API parameter to map to. Other XML standards may have similar elements.
5A		Identify and map each DATETIME element.
5B		Identify and map each AMOUNT element.
5C		Identify and map each OPERAMT element.
5D		Identify and map each QUANTITY element.
6		<b>ADD SIBLING FOR DUPLICATE NODES/ELEMENTS:</b> Add Siblings nodes/elements if needed.
7		<b>REVIEW DTD USERAREA:</b> Review all OAG's USERAREA definitions and determine whether they should be mapped to the Application Open Interface table column or Application API parameter. Other XML standards may have similar elements.
8		<b>RESOLVE DATA GAPS:</b> The DTD (source) elements for as many Application Open Interface (target) table columns or Application API parameters have been identified, and identify any data gaps.
9		<b>OPTIONAL Actions:</b> For Application products that use Application Open Interface tables to stage incoming data and utilize an Application Open Interface API to validate the staged data Use Insert to Database Table action to insert data into Application Open Interface tables Use Procedure Call action to execute Application Open Interface API For Application products that use application APIs that combine staging tables and the validation API. Use Procedure Call action and map source (DTD) data to the Application API parameter list.
10		<b>REQUIRED ACTION:</b> Add root-level, post-process action to raise business event indicating inbound process is complete. Add Procedure Call action for ECX_STANDARD.setEventDetails. See setEventDetails, page F-5 in the API Appendix.
11		<b>IDENTIFY SOURCE AND TARGET DOCUMENT LEVELS.</b>

## Compare Database Views (Source) to DTD (Target)

## Step 1 Create a Worksheet

Create a worksheet with the following columns: Target, Required (Y/N), Source, Code Category, Derivation Rule, Default Value, and Action, as shown in the following table:

Target	Required (Y/N)	Source	Code Category	Derivation Rule	Default Value	Action

## Step 2 Set Up Source

Get the DTD (source) for the required message or use a production XML message.

Include the the Field Identifier and Segment Descriptions, from the *Open Applications Group Integration Specification* Appendixes C and D, respectively.

**Note:** You may use third party software to expand the DTD or use a production XML message.

### Step 3 Set Up Target

Populate the target column of the worksheet with the Application Open Interface table definitions or Application API parameter list.

**Note:** You may use third party software to expand the DTD or use a production XML message.

### Step 3A Identify Required Target Elements

Identify the required Application Open Interface (target) table columns or Application API parameter list as follows:

- NOT NULL columns defined in the data model
- Application Open Interface API or Application API code to enforce required columns that are not defined as NOT NULL
- Open Interface or Application API documentation (for the product) identifying the required columns and associated derivation and defaulting rules.

### Step 4 Relate Source Element to Target Element

For each Application Open Interface (target) table column, identify the DTD (source) element to map from.

**Note:** Use the *Open Applications Group Integration Specification* Appendix C (Field Identifier Descriptions) and D (Segment Descriptions) to get detailed descriptions of each element identified on the DTD.

If the target column is for an internal ID, you must resolve this using a derivation or defaulting rule as outlined in steps 4B and 4C below. The sender cannot send an ID as they do not know what the valid internal IDs are. XML Gateway provides derivation actions to derive address and organization ID.

DTD attributes (lowercase tags) are not stored in the Oracle E-Business Suite, and therefore are not mapped. However, the value of the attribute can be used to determine the exact Application Open Interface table column or Application API parameter to map to.

#### **Step 4A**

If the Application Open Interface (target) table column can be sourced from a DTD (source) element, determine if the DTD (source) element value requires code conversion so the resulting value is meaningful to the Oracle E-Business Suite.

If code conversion is required, identify the code category from the list in Appendix B, page B-1.

Identify the from/to value to determine if the required values are available in the database you are using. If the values are not available, they must be added.

#### **Step 4B Identify Derivation Rule**

If the Application Open Interface (target) table column cannot be sourced from a DTD (source) element, determine if it can be derived. Refer to your product's Application Open Interface documentation for any predefined derivation rules. If a derivation rule is defined, make sure the incoming message provides the data to support the derivation rule because the Application Open Interface API will use the data to derive the value for the target column. Make a notation of the derivation rule on your worksheet so that you can verify this during unit testing.

If no derivation rule is defined, you can define one. The derivation rule can be based on several DTD elements or some default values. Make a notation of the derivation rule on your worksheet. Make a notation that an Action may be required.

Refer to Transaction Map - Actions, page 2-51 for a list of XML Gateway supported Actions.

Determine if the derivation rule (element or literal) requires code conversion for the trading partner. If code conversion is required, identify the code category from the list in Appendix B, page B-1.

Identify the from/to value to determine if the required values are available in the database. If the values are not available, they must be added.

#### **Step 4C Identify Defaulting Rule**

If the Application Open Interface (target) table column cannot be sourced from a DTD (source) element or derived, determine if it can be defaulted. Refer to your product's Application Open Interface documentation for any predefined defaulting rules. If a defaulting rule is defined, make sure the incoming message provides the data to support the defaulting rule. The Application Open Interface API will use the data as the default value for the target column. Make a notation of the defaulting rule on your worksheet so that you can verify this during unit testing.

If no defaulting rule is defined, you can define one. The default value may be set directly in the target definition using the Message Designer or as an XML Gateway Action if the default value is based on a condition. See Appendix C, page C-1 - XML Gateway Supported Actions.

Determine if the defaulting rule (element or literal) requires code conversion for the trading partner. If code conversion is required, identify the code category from the list in Appendix B, page B-1.

Identify the from/to value to determine if the required values are available in the database. If the values are not available, they must be added.

### **Step 5 Convert from OAG Format to Oracle Format**

For each DTD (source) DATETIME, AMOUNT, OPERAMT, or QUANTITY element, identify the Application Open Interface (target) column or Application API parameter to map to.

Oracle E-Business Suite represents date, amount, operating amount, and quantity in a single database column. OAG represents these as a collection of data. Oracle XML Gateway provides an Action to convert the OAG representation to the Oracle representation. Make a notation that this column requires an Action for "Convert from OAG" that will be defined using the Message Designer.

#### **Step 5A**

For the DATETIME element, identify the Application Open Interface (target) table column for date.

Values for attributes "qualifier," "type," and "index" are not mapped because the meaning is implied in the Application Open Interface table column for date.

**Note:** OAG DATETIME is made up of the following attributes and elements: qualifier, type, index, YEAR, MONTH, DAY, HOUR, M INUTE, SECOND, SUBSECOND, TIMEZONE.

#### **Step 5B**

For the AMOUNT element, identify the Application Open Interface (target) table columns for currency code and debit/credit flag if available. The XML Gateway "Convert from OAG" Action can be defined to store the OAG currency code and debit/credit flag in the columns identified, or ignored if the columns are not available.

Values for attributes "qualifier," "type," and "index" are not mapped because the meaning is implied in the Application Open Interface table column.

Values for NUMOFDEC and SIGN are implied in the Oracle representation of the amount value. The values are not mapped unless your application supports it. If so, you will need to map them using the Message Designer.

**Note:** OAG AMOUNT is made up of the following elements and attributes: qualifier, type, index, VALUE, NUMOFDEC, SIGN, CURRENCY, DRCCR.

#### **Step 5C**

For the OPERAMT element, identify the Application Open Interface (target) table columns for the currency code and unit of measure code if available. The XML Gateway "Convert from OAG" Action can be defined to store the OAG currency code and unit of measure code in the columns identified, or ignored if the columns are not available.

Values for attributes "qualifier" and "type" are not mapped because the meaning is implied in the Application Open Interface table column.

Values for NUMOFDEC, SIGN, UOMVALUE, and UOMNUMDEC are implied in the Oracle representation of the operating amount value. The values are not mapped unless your application supports it. If so, you will need to map them using the Message Designer.

**Note:** OAG OPERAMT is made up of the following attributes and elements: qualifier, type, VALUE, NUMOFDEC, SIGN, CURRENCY, UOMVALUE, UOMNUMDEC, UOM.

## Step 5D

For the QUANTITY element, identify the Application Open Interface (target) table column for the unit of measure code if available. The XML Gateway "Convert from OAG" Action can be defined to store the OAG unit of measure code in the column identified, or ignored if the column is not available.

The value for the "qualifier" attribute is not mapped because the meaning is implied in the Application Open Interface table column.

Values for NUMOFDEC and SIGN are implied in the Oracle representation of the quantity value. The values are not mapped unless your application supports it. If so, you will need to map them using the Message Designer.

**Note:** OAG QUANTITY is made up of the following attributes and elements: qualifier, VALUE, NUMOFDEC, SIGN, UOM.

## Step 6 Add Sibling for Duplicate Nodes/Elements

A DTD defines a single occurrence of an element, but additional occurrences can be added during message implementation to accommodate the business data. Map the duplicate node to the appropriate Application Open Interface table columns.

An example is the DTD element for PARTNER used to identify the trading party. If the Application Open Interface requires a SHIP-TO, BILL-TO, and REMIT-TO, then two PARTNER sibling nodes (same hierarchy level) were added to the original DTD to accommodate the BILL-TO and REMIT-TO. Map these entities to the appropriate Application Open Interface table columns.

## Step 7 Review DTD USERAREA

Consider that the original DTD may have been extended to support additional data required by the message. Review all USERAREA definitions and determine whether they should be mapped to the Application Open Interface table column or to an Application API parameter.

For outbound messages, all application flexfields were mapped to USERAREA fields. If the outbound message is being processed as an inbound message to another Oracle application module, the values may be meaningful. If so, they must be mapped to Application Open Interface table columns or Application API parameters.

## Step 8 Resolve Data Gaps

At this point, you have identified DTD (source) elements for as many Application Open Interface (target) table columns or Application API parameters as possible. Any required Application Open Interface (target) table column or Application API parameter that has not been mapped represents a data gap.



- If the required column is an internal ID, you must resolve this using a derivation or defaulting rule as described in Steps 4B and 4C, or use the Derive Address ID or Derive Parent ID actions.
- Consider resolving the data gaps by extending the DTD to include the required data. You can extend the DTD by adding sibling (same hierarchy level) or child (next level hierarchy) USERAREA elements or by adding new elements to an existing USERAREA. Refer to How to Extend DTDs, page 2-84 for information on how to define these.
- Consider creating new Application Open Interface tables and associated API. If this is a valid option, create and load the new table definitions into the database schema and create the message map.

**Note:** The only updates allowed for a DTD are to the USERAREA. Any other changes will invalidate the DTD.

### Step 9 Optional Actions

For Application products that use Application Open Interface tables to stage incoming data and utilize an Application Open Interface API to validate the staged data:

- Use the Insert to Database Table action to insert data into Application Open Interface tables.
- Use the Procedure Call action to execute Application Open Interface API.

For Application products that use application APIs that combine the staging tables and the validation API:

- Use the Procedure Call action and map source (DTD) data to the Application API parameter list. Refer to How to Map to an API, page 2-86 for additional instructions.

### Step 10 Required Action

Add root-level, post-process action to raise business event indicating inbound process is complete.

Add Procedure Call action for ECX\_STANDARD.setEventDetails. See setEventDetails, page F-5 in the API Appendix.

### Step 11 Identify Source and Target Document Levels

See Identify Source and Target Document Levels, page A-13.

## Identifying Source and Target Document Levels

After completing the map analysis, identify the document levels for both the source and target data.

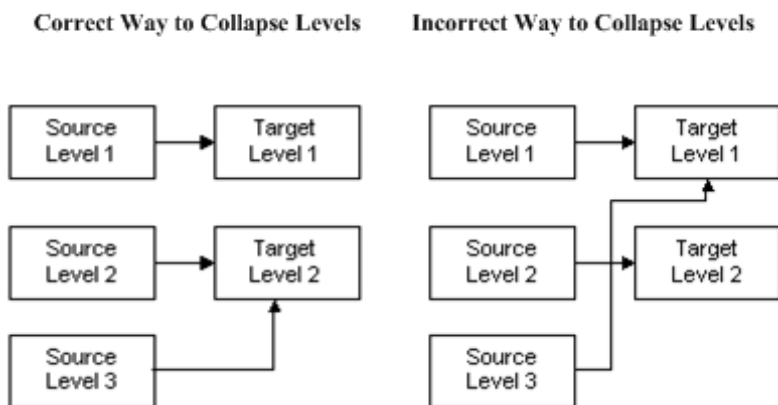
A document level represents a collection of data that repeats. For the Application Open Interface tables or database views, each table represents a document level. A level is the parent in a parent-child relationship.

DTDs do not use document levels because the levels are implied based on usage. Refer to the tree diagram in the OAG definitions to get a sense of how the data is grouped. The tree diagram will also give you an idea of where additional occurrences of a datatype may be required.

Once you have identified the document levels for the source and target, proceed to relate the source data structure to the target data structure. This task is straightforward if the number of data levels in the source and target are identical, but can be difficult if the numbers are different. If the number of levels of the source is greater than the target number, you must collapse levels. If the number of levels of the source is less than the target number, you must expand levels.

## Collapsing Levels

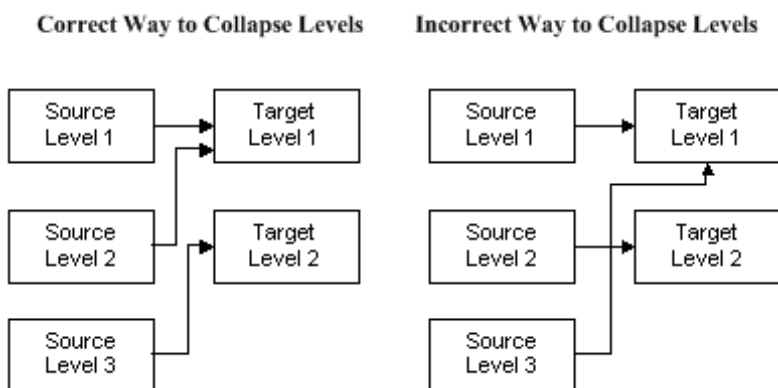
Collapsing levels is the mapping of multiple source levels to the same target level. For example, if your source is 3 levels and your target is 2 levels you can collapse the levels as shown in the following figure:



In the correct example above, the result of collapsing levels is that the data in Source Levels 2 and 3 are consolidated and mapped to Target Level 2. If there are two rows in Source Level 2 and three rows in Source Level 3, a total of six rows will be created in Target Level 2.

The incorrect example shows the collapsing of Source Levels 1 and 3 to Target Level 1, causing Source Level 3 to cross over Target Level 2.

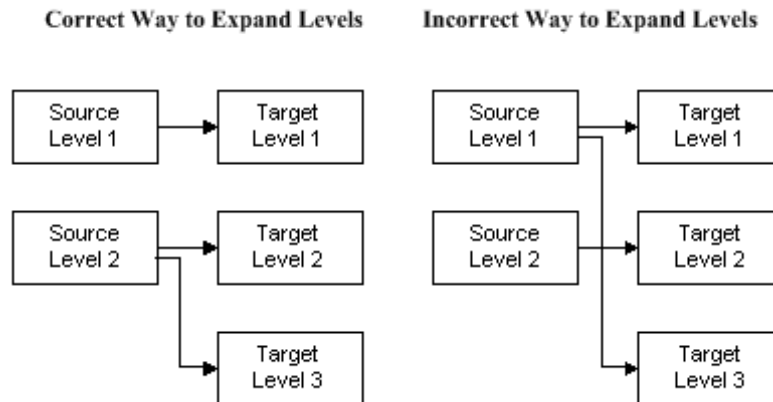
Another option is to relate Source Levels 1 and 2 to Target Level 1 and relate Source Level 3 to Target Level 2, as shown in the correct example below. (Do **not** map Source Level 3 to Target Level 1, crossing over Target Level 2.)



Whichever option you choose, consider what it means to promote lower level detail data to a higher level. The source data may need to be aggregated to be meaningful at the higher level.

## Expanding Levels

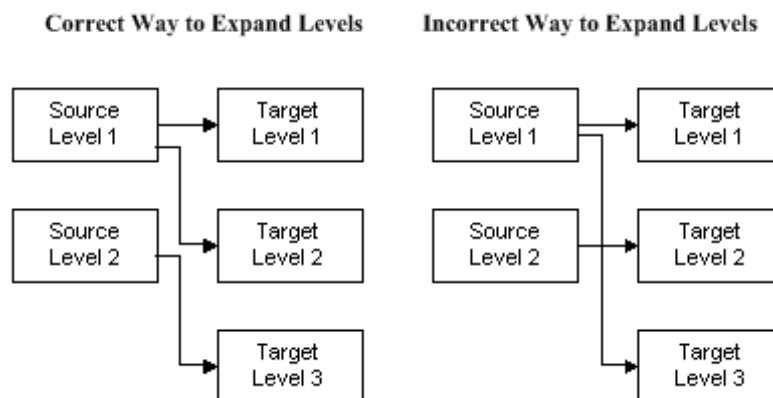
Expanding levels is the mapping of one source level to multiple target levels. For example, if your source is 2 levels and your target is 3 levels you can expand the levels as shown in the following figure:



The result of expanding levels, as shown in the correct example above, is that the data in Source Level 2 is distributed and mapped to Target Levels 2 and 3. If there are two rows in Source Level 2, two rows will be created in Target Level 2 and Target Level 3.

Do not expand Source Levels across Target Levels, as shown in the incorrect example above. Source Level 1 is incorrectly expanded to Target Levels 1 and 3, crossing over Target Level 2.

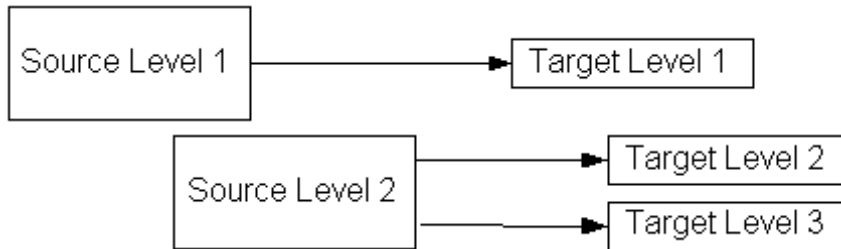
Another option, as shown in the correct example below, is to distribute Source Level 1 to Target Levels 1 and 2 and map Source Level 2 to Target level 3. (Do **not** map Source Level 1 to Target Levels 1 and 3, crossing over Target level 2, as shown in the incorrect example.)



Whichever option you choose, consider what it means to demote data from a higher level to a lower level of detail. The source data may need to be deaggregated to be meaningful at the lower level.

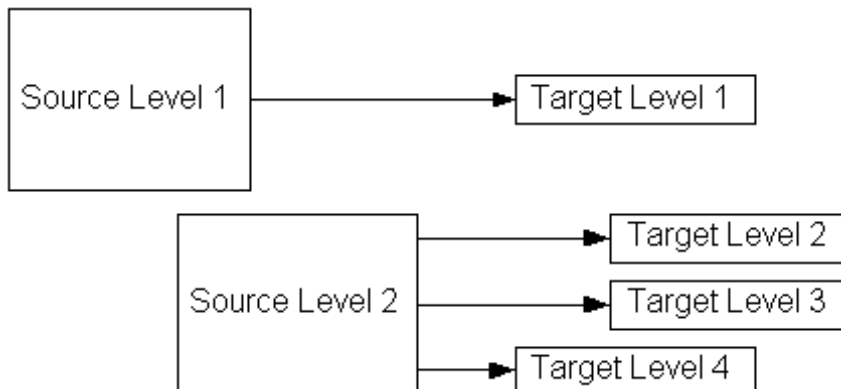
### Level Expansion for Discontinuous Nodes

Level expansion is supported if the target expanded levels are all siblings of each other or if they are all children of the previous node. The following diagram shows an example of correct expansion of levels.



In the example above, Target Level 2 and Target Level 3 are siblings to each other and children of Target Level 1.

The following diagram shows an example of invalid level expansion:



In the example of invalid level expansion above, Target Level 2 and Target Level 3 are siblings to each other and children of Target Level 1. Target Level 4 is a sibling of Target Level 1, with no relationship to Target Levels 2 and 3.

See Discontinuous Nodes, page 2-33 for more information on discontinuous nodes.

Another way to accommodate the data structure mismatch is to collapse or expand the database views or Application Open Interface tables and use the modified database views or application open interface table definitions to create the message map.

The resulting hierarchy mapping is defined using the Message Designer Level Mapping Tab.

## Recommending DTD Additions or Changes to OAG

Oracle is a member of the Open Application Group's standards committee chartered with developing industry-independent XML standards. This group meets periodically

to review recommendations and is responsible for publishing standards changes and additions.

During the map analysis process, you may discover that the needed fields are not represented in a DTD or that a needed DTD is not supported by OAG.

Whether you are recommending the creation of a brand new Business Object Document (BOD) and Document Type Definition (DTD) or additions or changes to an existing DTD, please present your recommendation in the OAG format and be prepared to identify the business case and justification. If you are making a recommendation to add or change an existing DTD, identify the DTD and version you are using. Please make sure your proposed DTD or DTD changes supports both the Application-to-Application (A2A) and Business-to-Business (B2B) scenarios.

If you would like Oracle to submit the request to OAG for you, enter an enhancement request. We will make the proposals to the OAG committee and report back to you. If you want to present your case directly to OAG, please let us know and we will include you in the meeting.

## **Special Considerations for Custom Messages**

Once a document is extracted from the Oracle E-Business Suite, consider updating the application tables to indicate the document has been extracted. This may require adding columns to the appropriate tables. This may be important to the Oracle E-Business Suite for document status reporting or to manage the supply and demand data.

Updating the Oracle E-Business Suite application tables was necessary in the EDI world to prevent a document from being extracted more than once. In the XML Gateway model, documents are extracted based on an event trigger in the Oracle E-Business Suite (as the documents are created) as opposed to batch processing used in the EDI model.

For inbound messages, consider adding calls to application procedures (as an Action) to link the Application Open Interface API process to the downstream process.



## Seeded Code Categories

### XML Gateway Seeded Code Categories

The XML Gateway seeded code categories and their descriptions are shown in the following table:

Category Code	Description
ACTION_CODE	Action Code (such as Add/Change/Delete)
AC_HANDL	Allowance and Charge Handling Code
AC_SPEC_CHARGES	Allowance and Charge Special Charges
AC_SPEC_SERVICES	Allowance and Charge Special Services
AETC_REASON	AETC Reason Code
AETC_RESPONSIBILITY	AETC Responsibility Code
AGREEMENT	Agreement Code
ALLOW_CHARGES_INDICATOR	Allowance or Charge Indicator Code
ALLOW_METHOD_HANDLING	Allowance or Charge Method of Handling Code
BARCODE	Bar Code
BUYER_ITEM	Buyer Item
CARRIER_CODE	Freight Carrier Code
CHARGE_ALLOW_QUALIFIER	Allowance or Charge Qualifier Code
CHARGE_TYPE	Charge Type Code
COMMODITY_COD_QUALIFIER	Commodity Code Qualifier Code
COMPANY_CODE	Company Code
CONTAINER_CODE	Container Code
CONTAINER_TYPE	Container Type Code
COUNTRY	Country Code
CURRENCY	Currency
CURRENCY_CODE	Currency Code

<b>Category Code</b>	<b>Description</b>
CURRENCY_CONV_TYPE	Currency Conversion Type Code
CUSTOMER_BANK_ACCOUNT	Customer Bank Account Code
CUSTOMER_BANK_ACCOUNT_TYPE	Customer Bank Account Type Code
CUSTOMER_BANK_BRANCH	Customer Bank Branch Code
CUSTOMER_BANK_BRANCH_ID	Customer Bank Branch ID
CUSTOMER_BANK_BRANCH_TYPE	Customer Bank Branch Type Code
CUSTOMER_BANK_LOCATION	Customer Bank Location Code
CUSTOMER_BANK_ORG	Customer Bank Org Code
CUSTOMER_BANK_SITE	Customer Bank Site Code
CUSTOMER_CODE	Customer Code
CUSTOMER_ITEM	Customer Item
CUSTOMER_LOCATION_CODE	Customer Location Code
CUSTOMER_NAME	Customer Name
CUSTOMER_NUMBER	Customer Number
CUSTOMER_ORG	Customer Org Code
CUSTOMER_SITE	Customer Site Code
CUSTOMER_SITE_ORG	Customer Site Organization Code
DATE_TIME_QUALIFIER	Date/Time Qualifier Code
DATE_TYPE	Date Type Code
DESTINATION_CLASS	Destination Class Code
DETAIL_SUBTYPE	Detail Subtype Code
DETAIL_TYPE	Detail Type Code
DOC_HANDLING_CODE	Document Handling Code
DOC_PURPOSE_CODE	Document Purpose Code
DOC_STATUS	Document Status Code
DOC_SUBTYPE	Document Subtype
DOC_TYPE	Document Type
EAN	European Article Number (EAN)
EQUIPMENT_CODE	Equipment Code
EQUIPMENT_DESC	Equipment Description Code
EXCESS_TRANS_REASON_CODE	Excess Transportation Reason Code



<b>Category Code</b>	<b>Description</b>
EXCESS_TRANS_RESPONSIBILITY	Excess Transportation Responsibility Code
EXPORT_DETAIL	Export Detail Code
EXPORT_HEADER	Export Header Code
FOB	Freight Code
FOB_PAYMENT	FOB Payment Code
FOB_POINT	FOB Point Code
FREIGHT_TERMS	Freight Terms Code
HAZ_CERTIFICATION	Hazardous Certification Code
HAZ_CLASS	Hazard Class Code
HAZ_MATERIAL	Hazardous Material Code
HAZ_MATL_CODE_QUAL	Hazardous Material Code Qualifier
HAZ_MATL_SHIPMENT_INFO_QUAL	Hazardous Shipment Info Qualifier
INTL_DOMESTIC_CODE	International/Domestic Code
INT_LOC_LOCATION_CODE	Internal Location Code
INT_LOC_ORG	Internal Location Org Code
INT_LOC_SITE	Internal Location Site Code
INVENTORY_ITEM	Inventory Item
INVOICE_TYPE	Invoice Type Code
ISBN_CODE	International Standard Book Number Code
ITEM_CATEGORY	Item Category Code
ITEM_QUALIFIER	Item Qualifier Code
ITEM_TYPE	Item Type Code
LABEL_NAME	Label Name Code
LADING_DESC_QUALIFIER	Lading Description Qualifier
METHOD_OF_PAYMENT	Method of Payment Code
NETWORKS	Network Code
PACKAGING_CODE	Packaging Code
PACKAGING_FROM	Packaging From Code
PACK_CODE	Pack Code
PAYMENT_FORMAT	Payment Format Code
PAYMENT_METHOD	Payment Method Code

<b>Category Code</b>	<b>Description</b>
PAYMENT_TERMS	Payment Terms Code
PAYMENT_TYPE	Payment Type Code
PRICE_BASIS	Price Basis Code
PRICE_IDENTIFIER	Price Identifier Code
PROVINCE	Province Code
PURCHASE_ORDER_TYPE	Purchase Order Type Code
QUALIFIER	Qualifier Code
QUANTITY_TYPE	Quantity Type Code
RATING_CATEGORY	Rating Category Code
RECORD_TYPE	Record Type Code
REFERENCE_CODE	Reference Code
REFERENCE_ID	Reference ID
REGION1	Region 1 Code
REGION2	Region 2 Code
REGION3	Region 3 Code
SALES_ORDER_TYPE	Sales Order Type Code
SCHEDULE_PURPOSE	Schedule Purpose Code
SCHEDULE_TYPE	Schedule Type Code
SEGMENT_DESC	Segment Description Code
SERVICE_LEVEL	Service Level Code
SHIPMENT_METHOD_PAYMENT	Shipment Method of Payment Code
SHIPMENT_PRIORITY	Shipment Priority Code
SHIPMENT_QUALIFIER	Shipment Qualifier Code
SHIPMENT_STATUS	Shipment Status Code
SHIP_DELIV_PATTERN	Shipment Delivery Pattern Code
SITE_QUALIFIER	Site Qualifier Code
SPECIAL_CHARGE_ALLOWANCE	Special Charge or Allowance Code
SPECIAL_HANDLING	Special Handling Code
SPECIAL_SERVICES	Special Services Code
STATE	State Code
SUPPLIER_BANK_ACCOUNT	Supplier Bank Account Code

<b>Category Code</b>	<b>Description</b>
SUPPLIER_BANK_ACCOUNT_TYPE	Supplier Bank Account Type Code
SUPPLIER_BANK_BRANCH	Supplier Bank Branch Code
SUPPLIER_BANK_BRANCH_ID	Supplier Bank Branch ID Code
SUPPLIER_BANK_BRANCH_TYPE	Supplier Bank Branch Type Code
SUPPLIER_BANK_LOCATION	Supplier Bank Location Code
SUPPLIER_BANK_ORG	Supplier Bank Org Code
SUPPLIER_BANK_SITE	Supplier Bank Site Code
SUPPLIER_CODE	Supplier Code
SUPPLIER_ITEM	Supplier Item Code
SUPPLIER_LOCATION_CODE	Supplier Location Code (such as EDI Location Code)
SUPPLIER_NAME	Supplier Name Code
SUPPLIER_NUMBER	Supplier Number Code
SUPPLIER_ORG	Supplier Organization Code
SUPPLIER_SCHEDULING_DESCRIPTOR	Supplier Scheduling Descriptor Code
SUPPLIER_SITE	Supplier Site Code
SUPPLIER_SITE_ORG	Supplier Site Organization Code
TAX_CODE	Tax Code
TAX_EXEMPT	Tax Exempt Code
TAX_EXEMPT_REASON	Tax Exempt Reason Code
TAX_JURISDICTION	Tax Jurisdiction Code
TAX_NAME	Tax Name Code
TERMS	Terms Code
TERMS_BASIS_DATE_CODE	Terms Basis Date Code
TERMS DUE DATE QUALIFIER	Terms Due Date Qualifier
TERMS_TYPE	Terms Type Code
TERRITORY	Territory Code
TRADING_PARTNER	Trading Partner Code
TRANSPORTATION_TERMS	Transportation Terms Code
TRANSPORTATION_TERMS_QUALIFIER	Transportation Terms Qualifier Code
UNIT_PRICE_BASIS	Basis of Unit Price Code
UOM	Unit of Measure Code

<b>Category Code</b>	<b>Description</b>
UPC	Uniform Product Code (UPC) Code
VESSEL_REQUIREMENT	Vessel Requirement Code
WAREHOUSE_CODE	Warehouse Code
WAREHOUSE_LOCATION_CODE	Warehouse Location Code
WAREHOUSE_NAME	Warehouse Name Code
WAREHOUSE_NUMBER	Warehouse Number Code
WAREHOUSE_ORG	Warehouse Organization Code
WAREHOUSE_SITE	Warehouse Site Code
WAREHOUSE_SITE_ORG	Warehouse Site Organization
WAYBILL REQUEST	Waybill Request Code

## Supported Actions

### XML Gateway Supported Actions

The following table lists the XML Gateway supported Actions. For the map analysis process, identify the required Action. For detailed descriptions of each Action and instructions on how to define an Action see Transaction Map -Actions, page 2-51.

Action Category	Action Description
Assignments	Assign variable value Create global value
Database Functions	Assign next sequence value Append where clause Insert into database table
Derivations	Derive Address ID from Location Code Derive Parent ID from Location Code
Function Call	Execute function and assign function return value
Math Functions	Add Divide Multiply Subtract
OAG Standard Conversions	Convert Oracle date to OAG date format Convert Oracle operating amount to OAG operating amount format Convert Oracle quantity to OAG quantity format Convert Oracle amount to OAG amount format Convert OAG date to Oracle date format Convert OAG operating amount to Oracle operating amount Convert OAG quantity to Oracle quantity format Convert OAG amount to Oracle amount format
Other	Exit program

Action Category	Action Description
Predefined Variable	Get predefined variable value: Code Conversion Return Status for (a specific element) Internal Control Number Return Code Return Message Receiver Trading Partner ID Sender Trading Partner ID Organization ID
Procedure Call	Execute procedure with send and return parameters
Return Error Message	Send error message to trading partner or XML Gateway system administrator contact
String Functions	Perform Concatenate Perform Substring
XSLT Transformation	Execute procedure to perform XSLT

Each ACTION may be based on a condition. A condition consists of two operands stated as a literal value or a variable. If a condition is defined, the ACTION is executed if the condition result is true. The condition operators supported by XML Gateway are listed in the following table:

Operator	Description
null	Null
not null	Not Null
=	Equal
!=	Not Equal
<	Less Than
>	Greater Than
<=	Less Than or Equal To
>=	Greater Than or Equal To

**Note:** Do not attempt to perform date comparisons unless your dates are stated as Julian dates. Use database functions to compare dates instead.

---

# Naming Conventions

## XML Gateway Naming Conventions Summary

There are several areas where naming conventions are defined to facilitate recognition of data. They are:

### Message Designer

- Message Map Names  
See Specify a Map Name, page 2-19.
- Data Definition Names  
See Select/Create a Source/Target Data Definition, page 2-20.
- XGM File Names  
See Transaction Map - Element Mapping, page 2-48.

### Setup Forms

- Transaction Type and Transaction Subtype  
See Define Transactions Form, page 3-7.
- External Transaction Type and External Transaction Subtype  
See Define Transactions Form, page 3-7.

### Processing

- Business Events  
See Integrating Oracle XML Gateway with Oracle Workflow Business Event System, page 6-1.





---

## Timezone Values

This appendix lists the valid time zone values for the profile option **ECX: Server Time Zone**.

This appendix covers the following topics:

- XML Gateway Valid Time Zone Values

### XML Gateway Valid Time Zone Values

Listed in the tables below are the valid values for the Applications profile option **ECX: Server Time Zone**. One of the following values must be entered exactly, or the time zone will default to Greenwich Mean Time (GMT).

The values are organized by geographical region. Select the region/city that corresponds to the time zone in which your database server is running.

For outbound transactions, the date and time data retrieved from the database, along with the time zone specified in the profile option, are used to determine the GMT deviation. The deviation is used in the XML message generated by the XML Gateway. No conversion is performed.

For inbound transactions, if the time zone of the incoming message is different from the time zone specified in the profile option, the incoming date and time will be converted.

The time zone values are categorized as follows:

- Africa, page E-2
- America, page E-3
- Antarctica, page E-6
- Asia, page E-6
- Atlantic, page E-8
- Australia, page E-8
- Europe, page E-9
- Indian, page E-10
- Pacific, page E-10

## Time Zone Values: Africa

---

### Time Zone Values: Africa

---

Africa/Abidjan

Africa/Accra

Africa/Addis\_Ababa

Africa/Algiers

Africa/Asmera

Africa/Bangui

Africa/Banjul

Africa/Bissau

Africa/Blantyre

Africa/Bujumbura

Africa/Cairo

Africa/Casablanca

Africa/Conakry

Africa/Dakar

Africa/Dar\_es\_Salaam

Africa/Djibouti

Africa/Douala

Africa/Freetown

Africa/Gaborone

Africa/Harare

Africa/Johannesburg

Africa/Kampala

Africa/Khartoum

Africa/Kigali

Africa/Kinshasa

Africa/Lagos

Africa/Libreville

Africa/Lome

Africa/Luanda

Africa/Lubumbashi

---

**Time Zone Values: Africa**

---

Africa/Lusaka  
Africa/Malabo  
Africa/Maputo  
Africa/Maseru  
Africa/Mbabane  
Africa/Mogadishu  
Africa/Monrovia  
Africa/Nairobi  
Africa/Ndjamena  
Africa/Niamey  
Africa/Nouakchott  
Africa/Ouagadougou  
Africa/Porto-Novo  
Africa/Sao\_Tome  
Africa/Timbuktu  
Africa/Tripoli  
Africa/Tunis  
Africa/Windhoek

---

**Time Zone Values: America**

---

**Time Zone Values: America**

---

America/Adak  
America/Anchorage  
America/Anguilla  
America/Antigua  
America/Aruba  
America/Asuncion  
America/Barbados  
America/Belize  
America/Bogota  
America/Buenos\_Aires

---

**Time Zone Values: America**

---

America/Caracas  
America/Cayenne  
America/Cayman  
America/Chicago  
America/Costa\_Rica  
America/Cuiaba  
America/Curacao  
America/Dawson\_Creek  
America/Denver  
America/Dominica  
America/Edmonton  
America/El\_Salvador  
America/Fortaleza  
America/Godthab  
America/Grand\_Turk  
America/Grenada  
America/Guadeloupe  
America/Guatemala  
America/Guayaquil  
America/Guyana  
America/Halifax  
America/Havana  
America/Indianapolis  
America/Jamaica  
America/La\_Paz  
America/Lima  
America/Los\_Angeles  
America/Managua  
America/Manaus  
America/Martinique  
America/Mazatlan

---

**Time Zone Values: America**

---

America/Mexico\_City  
America/Miquelon  
America/Montevideo  
America/Montreal  
America/Montserrat  
America/Nassau  
America/New\_York  
America/Noronha  
America/Panama  
America/Paramaribo  
America/Phoenix  
America/Port\_of\_Spain  
America/Port-au-Prince  
America/Porto\_Acre  
America/Puerto\_Rico  
America/Regina  
America/Santiago  
America/Santo\_Domingo  
America/Sao\_Paulo  
America/Scoresbysund  
America/St\_Johns  
America/St\_Kitts  
America/St\_Lucia  
America/St\_Thomas  
America/St\_Vincent  
America/Tegucigalpa  
America/Thule  
America/Tijuana  
America/Tortola  
America/Vancouver  
America/Winnipeg

---

## Time Zone Values: Antarctica

---

### Time Zone Values: Antarctica

---

Antarctica/Casey

Antarctica/DumontDURville

Antarctica/Mawson

Antarctica/McMurdo

Antarctica/Palmer

---

## Time Zone Values: Asia

---

### Time Zone Values: Asia

---

Asia/Aden

Asia/Alma-Ata

Asia/Amman

Asia/Anadyr

Asia/Aqtau

Asia/Aqtobe

Asia/Ashkhabad

Asia/Baghdad

Asia/Bahrain

Asia/Baku

Asia/Bangkok

Asia/Beirut

Asia/Bishkek

Asia/Brunei

Asia/Calcutta

Asia/Colombo

Asia/Dacca

Asia/Damascus

Asia/Dubai

Asia/Dushanbe

Asia/Hong\_Kong

Asia/Irkutsk

---

**Time Zone Values: Asia**

---

Asia/Ishigaki

Asia/Jakarta

Asia/Jayapura

Asia/Jerusalem

Asia/Kabul

Asia/Kamchatka

Asia/Karachi

Asia/Katmandu

Asia/Krasnoyarsk

Asia/Kuala\_Lumpur

Asia/Kuwait

Asia/Macao

Asia/Magadan

Asia/Manila

Asia/Muscat

Asia/Nicosia

Asia/Novosibirsk

Asia/Phnom\_Penh

Asia/Pyongyang

Asia/Qatar

Asia/Rangoon

Asia/Riyadh

Asia/Saigon

Asia/Seoul

Asia/Shanghai

Asia/Singapore

Asia/Taipei

Asia/Tashkent

Asia/Tbilisi

Asia/Tehran

Asia/Thimbu

---

**Time Zone Values: Asia**

---

Asia/Tokyo

Asia/Ujung\_Pandang

Asia/Ulan\_Bator

Asia/Vientiane

Asia/Vladivostok

Asia/Yakutsk

Asia/Yekaterinburg

Asia/Yerevan

---

## **Time Zone Values: Atlantic**

---

**Time Zone Values: Atlantic**

---

Atlantic/Azores

Atlantic/Bermuda

Atlantic/Canary

Atlantic/Cape\_Verde

Atlantic/Faeroe

Atlantic/Jan\_Mayen

Atlantic/Reykjavik

Atlantic/South\_Georgia

Atlantic/St\_Helena

Atlantic/Stanley

---

## **Time Zone Values: Australia**

---

**Time Zone Values: Australia**

---

Australia/Adelaide

Australia/Brisbane

Australia/Darwin

Australia/Lord\_Howe

Australia/Perth

Australia/Sydney

---



## Time Zone Values: Europe

---

### Time Zone Values: Europe

---

Europe/Amsterdam  
Europe/Andorra  
Europe/Athens  
Europe/Belgrade  
Europe/Berlin  
Europe/Brussels  
Europe/Bucharest  
Europe/Budapest  
Europe/Chisinau  
Europe/Copenhagen  
Europe/Dublin  
Europe/Gibraltar  
Europe/Helsinki  
Europe/Istanbul  
Europe/Kaliningrad  
Europe/Kiev  
Europe/Lisbon  
Europe/London  
Europe/Luxembourg  
Europe/Madrid  
Europe/Malta  
Europe/Minsk  
Europe/Monaco  
Europe/Moscow  
Europe/Oslo  
Europe/Paris  
Europe/Prague  
Europe/Riga  
Europe/Rome  
Europe/Samara

---

**Time Zone Values: Europe**

---

Europe/Simferopol

Europe/Sofia

Europe/Stockholm

Europe/Tallinn

Europe/Tirane

Europe/Vaduz

Europe/Vienna

Europe/Vilnius

Europe/Warsaw

Europe/Zurich

---

**Time Zone Values: Indian**

---

**Time Zone Values: Indian**

---

Indian/Antananarivo

Indian/Chagos

Indian/Christmas

Indian/Cocos

Indian/Comoro

Indian/Kerguelen

Indian/Mahe

Indian/Maldives

Indian/Mauritius

Indian/Mayotte

Indian/Reunion

---

**Time Zone Values: Pacific**

---

**Time Zone Values: Pacific**

---

Pacific/Apia

Pacific/Auckland

Pacific/Chatham

Pacific/Easter

---

**Time Zone Values: Pacific**

---

Pacific/Efate

Pacific/Enderbury

Pacific/Fakaofu

Pacific/Fiji

Pacific/Funafuti

Pacific/Galapagos

Pacific/Gambier

Pacific/Guadalcanal

Pacific/Guam

Pacific/Honolulu

Pacific/Kiritimati

Pacific/Kosrae

Pacific/Majuro

Pacific/Marquesas

Pacific/Nauru

Pacific/Niue

Pacific/Norfolk

Pacific/Noumea

Pacific/Pago\_Pago

Pacific/Palau

Pacific/Pitcairn

Pacific/Ponape

Pacific/Port\_Moresby

Pacific/Rarotonga

Pacific/Saipan

Pacific/Tahiti

Pacific/Tarawa

Pacific/Tongatapu

Pacific/Truk

Pacific/Wake

Pacific/Wallis

---



### XML Gateway APIs

The Oracle XML Gateway provides APIs for use with the Message Designer and for use by the execution engine.

- Execution engine-level APIs, page F-1
- Message Designer APIs, page F-3

### Execution Engine APIs

The following APIs are for use with the execution engine:

Package: ECX\_STANDARD, page F-1

perform\_xslt\_transformation, page F-1

Package: ECX\_ERRORLOG, page F-2

external\_system, page F-2

### APIs Defined in ECX\_STANDARD

#### perform\_xslt\_transformation

##### PL/SQL Syntax

```
procedure perform_xslt_transformation
(i_xml_file in out clob,
 i_xslt_file_name in varchar2,
 i_xslt_file_ver in number,
 i_xslt_application_code in varchar2,
 i_dtd_file_name in varchar2 default null,
 i_dtd_root_element in varchar2 default null,
 i_dtd_version in varchar2 default null,
 i_retcode out pls_integer,
 i_retmsg out varchar2);
```

##### Description

Used to apply a style sheet to an XML message and return the transformed XML messaged for further processing by the calling environment.

The DTD file name, version, and root element are required input parameters for this API, therefore the associated DTDs must be loaded into the XML Gateway repository. Refer to Loading and Deleting a DTD, page 2-88 for details on loading a DTD.

This API is independent of the Message Designer: XSLT Transformation action. This API is not intended for use in a message map.

**Note:** The profile option ECX\_XML\_VALIDATE\_FLAG must be set to "Y" for this action to be performed. For more information on this profile option, see Define System Profile Options, page 3-2.

#### Arguments (input)

**i\_xml\_file**

The XML message to be transformed.

**i\_xslt\_file\_name**

The XSLT style sheet file to be used for the transformation.

**i\_xslt\_file\_ver**

The version of the XSLT style sheet file. The highest version with the same file name and application code is used if no version number is provided.

**i\_xslt\_application\_code**

The name of the subdirectory where the XSLT style sheet is source-controlled (for example: ar/xml/xslt).

**i\_dtd\_file\_name**

The name of the DTD used in the XML message.

**i\_dtd\_root\_element**

The root element of the DTD used in the XML message.

**i\_dtd\_version**

Version of the DTD used in the XML message.

#### Arguments (output)

**i\_xml\_file**

The transformed XML message.

**i\_retcode**

Return code for the procedure.

**i\_retmsg**

Return message for the procedure.

## APIs Defined in ECX\_ERRORLOG

### external\_system

#### PL/SQL Syntax

```
procedure external_system
```

```

(i_outmsgid in raw,
 i_status in pls_integer,
 i_errmsg in varchar2,
 i_timestamp in date,
 o_ret_code out pls_integer,
 o_ret_msg out varchar2);

```

### Description

Used by both Oracle and non-Oracle messaging systems to report delivery status. The status information is written to the XML Gateway log tables to track and report transaction delivery data.

### Arguments (input)

#### **i\_outmsgid**

Message ID maintained by the XML Gateway execution engine for the outbound message delivered by the messaging system.

#### **i\_status**

Message delivery status as reported by the messaging system.

#### **i\_errmsg**

Error messages reported by the messaging system.

#### **i\_timestamp**

Time stamp from the messaging system indicating when it processed the outbound message created by XML Gateway.

### Arguments (output)

#### **o\_ret\_code**

Return code for the procedure

#### **o\_ret\_msg**

Return message for the procedure

## Message Designer APIs

XML Gateway has provided special purpose procedures and functions for use in the Message Designer to perform the following tasks:

1. Set event details to raise a business event for inbound transactions
2. Get event details using the event name maintained by the XML Gateway Execution Engine
3. Perform string manipulations
4. Get Trading Partner information using the sender's or receiver's Trading Partner ID maintained by the XML Gateway Execution Engine
5. Get message envelope data for inbound message
6. Get delivery data for outbound message
7. Get document logging information for a business document
8. Set message delivery status

9. Get System Administrator e-mail address from ECX: System Administrator Email Address system profile
10. Get sender logical ID from ECX\_OAG\_LOGICALID system profile
11. Set Error Exit

Use the Message Designer, Procedure Call or Function Call actions to initiate the APIs.

**Important:** With the exception of ECX\_ERRORLOG.external\_system, the procedures and functions in this appendix are meant for use in the Message Designer only and should not be used in any other context.

Most procedures include input and output arguments. You can map a source or target variable to the input arguments. The output arguments can be used in conjunction with the Send Error Message action for warnings or the ECX\_ACTIONS.set\_error\_exit\_program API for serious errors to send a notification. The notification may be sent to the Trading Partner contact, the system administrator (identified in the ECX: System Administrator Email Address system profile), or both.

If the successful completion of an API is critical to the success of your transaction, you may wish to use the Exit Program action to terminate the transaction if the API fails to process completely.

The following APIs are described in this section:

Package: ECX\_STANDARD, page F-1

setEventDetails, page F-5

getEventDetails, page F-14

getEventSystem, page F-7

getReferenceID, page F-8

Package: ECX\_DOCUMENT, page F-8

get\_delivery\_attribs, page F-8

Package: ECX\_CONDITIONS, page F-11

getLengthForString, page F-11

getPositionInString, page F-11

getSubString, page F-12

Package: ECX\_TRADING\_PARTNER\_PVT, page F-12

get\_receivers\_tp\_info, page F-12

get\_senders\_tp\_info, page F-13

get\_sysadmin\_email, page F-14

getEnvelopeInformation, page F-14

getOAGLOGICALID, page F-16

Package: ECX\_ERRORLOG, page F-17

getDoclogDetails, page F-17

external\_system, page F-2



Package: ECX\_ACTIONS, page F-19  
set\_error\_exit\_program, page F-19  
Package: ECX\_ATTACHMENT, page F-20  
register\_attachment, page F-20  
retrieve\_attachment, page F-22  
reconfig\_attachment, page F-23  
Package: ECX\_ENG\_UTILS, page F-25  
convert\_to\_cxml\_date, page F-25  
convert\_to\_cxml\_datetime, page F-25  
convert\_from\_cxml\_datetime, page F-26

## APIs Defined in ECX\_STANDARD

### setEventDetails

#### PL/SQL Syntax

```
procedure setEventDetails
(eventname in varchar2,
 eventkey in varchar2,
 parameter1 in varchar2,
 parameter2 in varchar2,
 parameter3 in varchar2,
 parameter4 in varchar2,
 parameter5 in varchar2,
 parameter6 in varchar2,
 parameter7 in varchar2,
 parameter8 in varchar2,
 parameter9 in varchar2,
 parameter10 in varchar2,
 retcode out pls_integer,
 retmsg out varchar2);
```

#### Description

Sets event details to raise a business event for inbound transactions.

This is defined at the root level as a post-process action to indicate that an inbound message has been processed. Any event subscription defined in the Oracle e-Business Suite interested in this inbound message will proceed to consume it.

This procedure must be used for all inbound transactions. The argument values vary by transaction with the exception of the confirmation message where the argument values are specific. See How to Implement an OAG Confirmation BoD, page 4-13 to see how the event details are defined.

#### Arguments (input)

##### **eventname**

Unique identifier for the business event associated with the inbound message. The event name consists of the following components:

ORACLE.APPS.<COMPONENT>.<TASK>.<EVENT>

where:

COMPONENT is based on the internal transaction type entered using the Define Transactions window. It represents the product short code.

TASK is based on the internal transaction subtype entered using the Define Transactions window. It represents a description of the object.

EVENT is a literal that describes the business function of the message.

Following is an example of an Event Name that identifies a confirmation event associated with an outbound purchase order ORACLE.APPS.PO.POO.CONFIRM

**eventkey**

Unique identifier for the business document from the Oracle e-Business Suite associated with the business event.

**parameter1 through  
parameter10**

User-defined parameters to pass data of interest to the event subscription defined in the Oracle e-Business Suite for the inbound business document.

**Arguments (output)**

**retcode**

Return code for the procedure.

**retmsg**

Return message for the procedure.

**getEventDetails**

**PL/SQL Syntax**

```
procedure getEventDetails
(eventname out varchar2,
 eventkey out varchar2,
 itemtype out varchar2,
 itemkey out varchar2,
 parentitemtype out varchar2,
 parentitemkey out varchar2,
 retcode out pls_integer,
 retmsg out varchar2);
```

**Description**

Gets event details using the event name maintained by the XML Gateway Execution Engine.

**Arguments (input)**

None.

**Arguments (output)**

**eventname**

Event name passed internally to the procedure. It is a unique identifier for the business event associated with the transaction.

**eventkey**

Event key associated with the event name passed.

**itemtype**

Unique identifier for a group of objects that share the same set of item attributes (also known as variables).

Item types are created using the Workflow Builder and are used by Oracle e-Business Suite application modules to group related functions.

**itemkey**

Unique identifier for an item in an item type

**parentitemtype**

Parent item type for the item type

**parentitemkey**

Parent item key for the item key

**retcode**

Return code for the procedure

**retmsg**

Return message for the procedure

**getEventSystem****PL/SQL Syntax**

```
(from_agent out varchar2,
 to_agent out varchar2,
 from_system out varchar2,
 to_system out varchar2,
 retcode out pls_integer,
 retmsg out varchar2);
```

**Description**

Gets event details related to the system and agent using the event name maintained by the XML Gateway Execution Engine.

The procedure is context-sensitive, so you will receive return values relevant to the context. The "from" parameters are for inbound transactions. The "to" parameters are for outbound transactions.

**Arguments (input)**

None

**Arguments (output)****from\_agent**

Workflow agent (queue) the inbound message is dequeued from.

**to\_agent**

Workflow agent (queue) the outbound message is enqueued to.

**from\_system**

System processing the inbound message

**to\_system**

System processing the outbound message

**retcode**

Return code for the procedure

**retmsg**

Return code for the message

**getReferenceID****PL/SQL Syntax**

```
procedure getReferenceID  
return out varchar2;
```

**Description**

Returns the value associated with the REFERENCEID element of the OAG CONTROLAREA. The field contains a concatenated value consisting of system name, event name, and event key delimited by ":". This procedure is used for message maps in which the OAG standard is not used.

For message maps created using the OAG standard, the ECX\_OAG\_CONTROLAREA\_T P\_V view is used to retrieve the reference\_id identified by the business event. This value is used to map to the OAG CONTROLAREA, REFERENCEID element.

**Arguments (input)**

None.

**Arguments (output)**

None.

**APIs Defined in ECX\_DOCUMENT****get\_delivery\_attribs****PL/SQL Syntax**

```
procedure get_devlivery_attribs
```

```

(transaction_type in varchar2,
 transaction_subtype in varchar2,
 party_id in varchar2,
 party_site_id in varchar2,
 party_type in/out varchar2,
 standard_type out varchar2,
 standard_code out varchar2,
 ext_type out varchar2,
 ext_subtype out varchar2,
 source_code out varchar2,
 destination_code out varchar2,
 destination_type out varchar2,
 destination_address out varchar2,
 username out varchar2,
 password out varchar2,
 map_code out varchar2,
 queue_name out varchar2,
 tp_header_id out pls_integer,
 retcode out pls_integer,
 retmsg out varchar2);

```

## Description

Gets setup data using the internal transaction type, subtype, party id, party\_type, and party site id. The data is required to process an outbound transaction.

## Arguments (input)

### **transaction\_type**

Internal transaction type passed to the procedure. The internal transaction type is entered using the Define Transactions window.

### **transaction\_subtype**

Internal transaction subtype passed to the procedure. The internal transaction subtype is entered using the Define Transactions window.

### **party\_id**

Trading partner ID passed to the procedure. Party site ID will be used if Party ID is null. The trading partner ID is entered using the Define Trading Partners window.

### **party\_site\_id**

Trading partner site ID passed to the procedure. The trading partner site ID is entered using the Define Trading Partners window.

### **party\_type**

Party type associated with the trading partner and trading partner site passed to the procedure. Party type is entered using the Define Trading Partners window.

## Arguments (output)

### **party\_type**

Party type associated with the trading partner and trading partner site passed to the procedure. Party type is entered using the Define Trading Partners window.

### **standard\_type**

The XML standard type associated with the transaction passed to the procedure. Standard type is entered using the Define XML Standards window.

**standard\_code**

The standard code (for example, OAG) associated with the transaction passed to the procedure. Standard code is entered using the Define XML Standards window.

**ext\_type**

External transaction type associated with the internal transaction type passed to the procedure. External transaction type is entered using the Define Transactions window.

**ext\_subtype**

External transaction subtype associated with the internal transaction subtype passed to the procedure. External transaction subtype is entered using the Define Transactions window.

**source\_code**

Source location code associated with the trading partner and trading partner site passed to the procedure. Source location code is entered using the Define Trading Partners window.

**destination\_code**

Destination location code associated with the trading partner and trading partner site passed to the procedure. Destination location code is entered using the Define Trading Partners window.

**destination\_type**

Destination location type associated with the trading partner and trading partner site passed to the procedure. Destination location type is entered using the Define Trading Partners window.

**destination\_address**

Destination address associated with the trading partner and trading partner site passed to the procedure. Destination address is entered using the Define Trading Partners window.

**username**

Username associated with the trading partner and trading partner site passed to the procedure. Username is entered using the Define Trading Partners window.

**password**

Password associated with the username for the trading partner and trading partner site passed to the procedure. The password is returned in encrypted format. Password is entered using the Define Trading Partners window.

**map\_code**

Message map associated with the trading partner and transaction passed to the procedure. The map code is entered using the Define Trading Partners window.

**queue\_name**

Queue name associated with the transaction passed to the procedure. The queue name is entered using the Define Transactions window.

**tp\_header\_id**

System-generated identifier for the trading partner passed to the procedure, entered using the Define Trading Partners window

**retcode**

Return code for the procedure

**retmsg**

Return message for the procedure

## APIs Defined in ECX\_CONDITIONS

### getLengthForString

#### PL/SQL Syntax

```
procedure getLengthForString
(i_string in varchar2);
  i_length out pls_integer);
```

#### Description

Determines the length of the string passed to it. This procedure may be used in conjunction with the getPositionInString procedure or the Perform Substring action.

#### Arguments (input)

**i\_string**  
The input string.

#### Arguments (output)

**i\_length**  
The length of the input string.

### getPositionInString

#### PL/SQL Syntax

```
procedure getPositionInString
(i_string in varchar2,
 i_search_string in varchar2,
 i_start_position in pls_integer, default null,
 i_occurrence in pls_integer, default null,
 i_position out pls_integer);
```

#### Description

Parses a concatenated string with delimiters into its individual components. The **i\_search\_string** parameter identifies the delimiter. The **i\_occurrence** parameter identifies which occurrence of the delimiter to check for. The return value of the procedure is the position of the first character of the portion of the string you are interested in.

This procedure can be used in conjunction with the getLengthForString procedure or the Perform Substring action.

#### Arguments (input)

**i\_string**  
The input string.

**i\_search\_string**  
The delimiter used in the concatenated string.

**i\_start\_position**  
The character position to begin parsing from.

**i\_occurrence**

The occurrence of the delimiter to search for.

**Arguments (output)****i\_position**

The character position of the first character of the portion of the input string you are interested in.

**getSubString****PL/SQL Syntax**

```
procedure getSubString
(i_string in varchar2,
 i_start_position in pls_integer, default 0,
 i_length in pls_integer, default 0,
 i_substr out varchar2);
```

**Description**

Parses a string passed to it given the start position and the length of the substring.

This procedure is used when the length of the substring is maintained in a variable whereas the Perform Substring action is used if the length of the string is a literal value.

**Arguments (input)****i\_string**

The input string.

**i\_start\_position**

The character position to begin parsing from.

**i\_length**

Length from start position of input string to include in resulting substring.

**Arguments (output)****i\_substr**

The substring.

**APIs Defined in ECX\_TRADING\_PARTNER\_PVT****get\_receivers\_tp\_info****PL/SQL Syntax**

```
procedure get_receivers_tp_info
(p_party_id out number,
 p_party_site_id out number,
 p_org_id out pls_integer,
 p_admin_email out varchar2,
 retcode out pls_integer,
 retmsg out varchar2);
```



**Description**

Gets Trading Partner data using the receiver's trading partner ID maintained by the XML Gateway Execution Engine. The Trading Partner data is entered using the Define Trading Partners window.

**Arguments (input)**

None

**Arguments (output)****p\_party\_id**

Trading Partner ID

**p\_party\_site\_id**

Site associated with the Trading Partner.

**p\_org\_id**

Organization associated with the Trading Partner site.

**p\_admin\_email**

E-mail address associated with the Trading Partner contact.

**retcode**

Return code for the procedure

**retmsg**

Return message for the procedure

**get\_senders\_tp\_info****PL/SQL Syntax**

```
procedure get_senders_tp_info
(p_party_id out number,
 p_party_site_id out number,
 p_org_id out pls_integer,
 p_admin_email out varchar2,
 retcode out pls_integer,
 retmsg out varchar2);
```

**Description**

Gets Trading Partner data using the sender's trading partner ID maintained by the XML Gateway Execution Engine. The Trading Partner data is entered using the Define Trading Partners window.

**Arguments (input)**

None

**Arguments (output)****p\_party\_id**

Trading Partner ID

**p\_party\_site\_id**

Site associated with the Trading Partner.

**p\_org\_id**

Organization associated with the Trading Partner site.

**p\_admin\_email**

E-mail address associated with the Trading Partner contact.

**retcode**

Return code for the procedure

**retmsg**

Return message for the procedure

## **get\_sysadmin\_email**

### **PL/SQL Syntax**

```
procedure get_sysadmin_email
(email_address out varchar2,
 retcode out pls_integer,
 errmsg out varchar2);
```

### **Description**

Gets the system administrator e-mail address defined for the ECX: System Administrator Email Address profile option.

This procedure is not required if you are using the Send Error Message action to send notifications to the system administrator. The address for the system administrator is derived.

### **Arguments (input)**

None

### **Arguments (output)**

**email\_address**

E-mail address identified in the ECX: System Administrator Email Address system profile.

**retcode**

Return code for the procedure

**errmsg**

Return message for the procedure

## **getEnvelopeInformation**

### **PL/SQL Syntax**

```
procedure getEnvelopeInformation
```

```

(i_internal_control_number in pls_integer,
 i_message_type out varchar2,
 i_message_standard out varchar2,
 i_transaction_type out varchar2,
 i_transaction_subtype out varchar2,
 i_document_number out varchar2,
 i_party_id out varchar2,
 i_party_site_id out varchar2,
 i_protocol_type out varchar2,
 i_protocol_address out varchar2,
 i_username out varchar2,
 i_password out varchar2,
 i_attribute1 out varchar2,
 i_attribute2 out varchar2,
 i_attribute3 out varchar2,
 i_attribute4 out varchar2,
 i_attribute5 out varchar2,
 retcode out pls_integer,
 retmsg out varchar2);

```

## Description

Retrieves message envelope data using the internal control number maintained by the XML Gateway Execution Engine.

See XML Gateway Envelope, page 4-6 for information regarding the message envelope.

## Arguments (input)

### **i\_internal\_control\_number**

Internal control number maintained by the XML Gateway execution engine associated with the inbound message.

## Arguments (output)

### **i\_message\_type**

The message type is defaulted to "XML"

### **i\_message\_standard**

The XML standard used for the business document received from the Trading Partner.

The XML standard is entered using the Define XML Standards window and used in the Define Transactions window.

### **i\_transaction\_type**

External transaction type associated with the business document received from the Trading Partner.

The external transaction type is entered using the Define Transactions window and is used in the Define Trading Partners window.

### **i\_transaction\_subtype**

External transaction subtype associated with the business document received from the Trading Partner.

The external transaction subtype is entered using the Define Transactions window and is used in the Define Trading Partners window.

**i\_document\_number**

Unique identifier for the business document received from the Trading Partner. This field is not used by XML Gateway but is available for the Oracle e-Business Suite receiving application module.

**i\_party\_id**

Not Used.

**i\_party\_site\_id**

The source Trading Partner Location Code entered using the Define Trading Partners window if no data is found in the Destination Trading Partner Location Code.

**i\_protocol\_type**

The transmission method entered using the Define Trading Partners window.

**i\_protocol\_address**

The address/URL associated with the transmission method. It is entered using the Define Trading Partners window.

**i\_username**

The username entered using the Define Trading Partners window.

**i\_password**

The password associated with the username. Entered using the Define Trading Partners window.

**i\_attribute1**

User-defined field to pass data.

**i\_attribute2**

User-defined field to pass data.

**i\_attribute3**

Data in this field will trigger the creation of another XML message that is sent to the Trading Partner identified in the Destination Trading Partner Location Code field entered using the Define Trading Partners window.

**i\_attribute4**

User-defined field to pass data.

**i\_attribute5**

User-defined field to pass data.

**retcode**

Return code for the procedure

**retmsg**

Return message for the procedure

**getOAGLOGICALID****PL/SQL Syntax**

```
procedure getOAGLOGICALID
return varchar2;
```

**Description**

Gets the sender's logical ID defined for the ECX\_OAG\_LOGICALID system profile.

The value defined in the ECX\_OAG\_LOGICALID system profile is retrieved by the ECX\_OAG\_CONTROLAREA\_TP\_V view. The value is used to map to the OAG CNTROLAREA, LOGICALID element.

This function is not required if you are using the ECX\_OAG\_CONTROLAREA\_TP\_V view.

**Arguments (input)**

None

**Arguments (output)**

None

## APIs Defined in ECX\_ERRORLOG

### getDoclogDetails

**PL/SQL Syntax**

```
procedure getDoclogDetails
(i_msgid in raw,
 i_message_type out varchar2,
 i_message_standard out varchar2,
 i_transaction_type out varchar2,
 i_transaction_subtype out varchar2,
 i_document_number out varchar2,
 i_party_id out varchar2,
 i_party_site_id out varchar2,
 i_protocol_type out varchar2,
 i_protocol_address out varchar2,
 i_username out varchar2,
 i_password out varchar2,
 i_attribute1 out varchar2,
 i_attribute2 out varchar2,
 i_attribute3 out varchar2,
 i_attribute4 out varchar2,
 i_attribute5 out varchar2,
 i_logfile out varchar2,
 i_internal_control_number out number,
 i_status out varchar2,
 i_time_stamp out date,
 i_direction out varchar2,
 retcode out pls_integer,
 retmsg out varchar2);
```

**Description**

Gets information about transactions processed by XML Gateway. An entry is written to the ECX\_DOCLOGS table for each outbound message created by XML Gateway and each inbound message processed by XML Gateway.

Error recovery is performed using the stored copy of a message. For details regarding error recovery, see XML Gateway Error Processing Item Type, page 6-5.

### Arguments (input)

**i\_msgid**

Message identifier provided by the XML Gateway execution engine for each message processed.

### Arguments (output)

**i\_message\_type**

The message type is defaulted to "XML"

**i\_message\_standard**

The XML standard associated with the business document as entered using the Define XML Standards window and used by the Define Trading Partners window.

**i\_transaction\_type**

External transaction type associated with the business document as entered using the Define Transactions window and used by the Define Trading Partners window.

**i\_transaction\_subtype**

External transaction subtype associated with the business document as entered using the Define Transactions window and used by the Define Trading Partners window.

**i\_document\_number**

Unique identifier from the Oracle e-Business Suite for the outbound business document.

Unique identifier for an inbound business document received from the Trading Partner.

**i\_party\_id**

Trading Partner identifier associated with the business document as entered using the Define Trading Partners window.

**i\_party\_site\_id**

Trading Partner site identifier associated with the business document as entered using the Define Trading Partners window.

**i\_protocol\_type**

Communication method associated with the business document as entered using the Define Trading Partners window.

**i\_protocol\_address**

Address/URL associated with the communication method as entered using the Define Trading Partners window.

**i\_username**

The username associated with the Trading Partner as entered using the Define Trading Partners window.

**i\_password**

The password associated with the username as entered using the Define Trading Partners window.

**i\_attribute1**

User-defined field to pass data.

**i\_attribute2**

User-defined field to pass data.

**i\_attribute3**

The Destination Trading Partner Location Code as entered using the Define Trading Partners window.

**i\_attribute4**

User-defined field to pass data.

**i\_attribute5**

User-defined field to pass data.

**i\_logfile**

Identifies the name of the log file created by the XML Gateway execution engine for the business document processed. The log files are written to the directory identified in the ECX: Log File Path system profile.

**i\_internal\_control\_  
number**

The unique identifier as defined by the XML Gateway execution engine for each business document processed.

**i\_status**

The status of the business document processed.

**i\_time\_stamp**

Time stamp from the XML Gateway execution engine representing when the outbound business document was created or when the inbound business document was processed.

**i\_direction**

Identifies whether the business document was outbound from the Oracle e-Business Suite or inbound into the Oracle e-Business Suite.

**retcode**

Return code for the function.

**retmsg**

Return message for the function.

## APIs Defined in ECX\_ACTIONS

### set\_error\_exit\_program

#### PL/SQL Syntax

```
procedure set_error_exit_program
(i_err_type in pls_integer,
 i_err_code in pls_integer,
 i_err_msg in varchar2);
```

#### Description

Sets the error code and message so that a notification can be sent to either the Trading Partner contact, the System Administrator (identified in the ECX: System Administrator Email Address system profile), or both.

The Send Error Message action is used for warnings that do not require the process to be terminated. This procedure is used for serious errors where you intend to terminate the process.

For more details regarding the Send Error Message action, see Map Action Editor - Return Error Message: Send Error Message, page 2-81.

#### Arguments (input)

**i\_err\_type**

A code to identify the intended recipient of the error notification. The valid values are as follows:

10 = Reporting error, do not send notification

20 = Send notification to the Trading Partner contact containing the error code and message

25 = Send notification to both the Trading Partner and System Administrator contact containing the error code and message

30 = Send notification to the System Administrator contact containing the error code and message

**i\_err\_code**

Code for error detected

**i\_err\_msg**

Message string for errors detected. Multiple messages may be concatenated into this variable and sent in the notification.

#### Arguments (output)

None.

### APIs Defined in ECX\_ATTACHMENT

#### register\_attachment

##### PL/SQL Syntax

```
procedure register_attachment
(i_entity_name in varchar2,
 i_pk1_value in varchar2,
 i_pk2_value in varchar2,
 i_pk3_value in varchar2,
 i_pk4_value in varchar2,
 i_pk5_value in varchar2,
 i_file_id in number,
 i_data_type in number,
 x_cid out varchar2);
```

#### Description

Called by message maps for outbound documents to register the correlation of attachment(s) to an outbound business document.

This API assumes the attachment has been defined in the Oracle Foundation module. The required input to the API are values returned when the attachment was deposited in FND.

#### Arguments (input)

**i\_entity\_name**

Entity name from the FND\_LOBS table used to compose the x\_cid value.



**i\_pk1\_value**

Key value from the FND\_LOBS table used to compose the x\_cid value.

**i\_pk2\_value**

Key value from the FND\_LOBS table used to compose the x\_cid value.

**i\_pk3\_value**

Key value from the FND\_LOBS table used to compose the x\_cid value.

**i\_pk4\_value**

Key value from the FND\_LOBS table used to compose the x\_cid value.

**i\_pk5\_value**

Key value from the FND\_LOBS table used to compose the x\_cid value.

**i\_file\_id**

File identifier from the FND\_LOBS table used to determine the file\_name value from the FND\_LOBS table. The file\_name is used to compose the x\_cid value.

**i\_data\_type**

Identifies the data type of the attachment file. The valid value is BLOB. Currently, only the ecx\_attachment.embedded\_lob\_data\_type value is supported (only

**Arguments (output)****x\_cid**

A unique identifier for the attachment within a given outbound document provided by FND when the attachment was deposited. The value is constructed using the formulate\_content\_id API by concatenating i\_entity\_name, i\_pk1\_value, i\_pk2\_value, i\_pk3\_value, i\_pk4\_value, i\_pk5\_value (which are provided as input to this API), and then adding the file\_name (based on the file\_id) from the FND\_LOBS table.

**Note:** register\_attachment is an overloaded API. The second signature is documented below. The difference between the two APIs is that the first uses the key values from the FND\_LOBS table to uniquely identify the attachment; and the second uses a user-defined identifier.

**register\_attachment****PL/SQL Syntax**

```
procedure register_attachment
(i_cid in varchar2,
 i_file_id in number,
 i_data_type in number);
```

**Arguments (input)****i\_cid**

A unique identifier for the attachment provided by the user when the attachment was defined in FND.

**i\_file\_id**

File identifier from the FND\_LOBS table.

**i\_data\_type**

Identifies the data type of the attachment file. The valid value is BLOB. The list of valid values is maintained in ecx\_attachment.embedded\_lob\_data\_type API.

### Arguments (output)

None.

## retrieve\_attachment

### PL/SQL Syntax

```
procedure retrieve_attachment
(i_msgid in raw,
 x_cid in varchar2,
 x_file_name out varchar2,
 x_file_content_type out varchar2,
 x_file_data out nocopy blob,
 x_ora_charset out varchar2,
 x_file_format out varchar2);
```

### Description

Called by message maps for inbound documents to retrieve an attachment deposited by Oracle Transport Agent (OTA) when the inbound document was received.

Not all attachments deposited by OTA are of interest to the receiving application. Only the attachments identified by the `i_msgid` and `x_cid` parameters are retrieved using this API.

### Arguments (input)

#### **i\_msgid**

The message ID associated with the attachment deposited by OTA into the FND repository.

#### **x\_cid**

A unique identifier for the attachment provided in the inbound XML document. With OAG, this would be provided in the ATTCHREF data type, FILENAME element. The exact location of the unique identifier in the XML document will vary by standard.

### Arguments (output)

#### **x\_file\_name**

Name of the attachment file from the FND\_LOBS table.

#### **x\_file\_content\_type**

Content type specified during the attachment uploading process. Information from the FND\_LOBS table.

#### **x\_file\_data**

The uploaded attachment stored as a binary LOB from the FND\_LOBS table.

#### **x\_ora\_charset**

Oracle character set from the FND\_LOBS table.

#### **x\_file\_format**

File format ("text" or "binary") from the FND\_LOBS table.

## reconfig\_attachment

### PL/SQL Syntax

```
procedure reconfig_attachment
(i_msgid in raw,
 i_cid in varchar2,
 i_entity_name in varchar2,
 i_pk1_value in varchar2,
 i_pk2_value in varchar2,
 i_pk3_value in varchar2,
 i_pk4_value in varchar2,
 i_pk5_value in varchar2,
 i_program_app_id in number,
 i_program_id in number,
 i_request_id in number,
 x_document_id out number);
```

### Description

Called by message maps for inbound documents to reset FND attributes for a previously retrieved attachment deposited by OTA when the inbound document was received.

Not all attachments retrieved using the retrieve\_attachment API must be reconfigured. Use the reconfig\_attachment API only if you want to update the FND attributes. Use the standard Oracle Foundation module to create a new copy of the attachment if necessary.

### Arguments (input)

#### **i\_msgid**

The message ID associated with the attachment deposited by OTA into the FND repository.

#### **i\_cid**

A unique identifier for the attachment. This is the same value provided to the retrieve\_attachment API that was based on the value in the inbound XML document.

#### **i\_entity\_name**

Entity name from the FND\_ATTACHED\_DOCUMENTS table.

#### **i\_pk1\_value**

Key value from the FND\_ATTACHED\_DOCUMENTS table.

#### **i\_pk2\_value**

Key value from the FND\_ATTACHED\_DOCUMENTS table.

#### **i\_pk3\_value**

Key value from the FND\_ATTACHED\_DOCUMENTS table.

#### **i\_pk4\_value**

Key value from the FND\_ATTACHED\_DOCUMENTS table.

#### **i\_pk5\_value**

Key value from the FND\_ATTACHED\_DOCUMENTS table.

#### **i\_program\_app\_id**

Standard extended who column from FND\_ATTACHED\_DOCUMENTS table.

**i\_program\_id**

Standard extended who column from FND\_ATTACHED\_DOCUMENTS table.

**i\_request\_id**

Standard extended who column from FND\_ATTACHED\_DOCUMENTS table.

**Arguments (output)****x\_document\_id**

Unique identifier for the reconfigured attachment.

**formulate\_content\_id****PL/SQL Syntax**

```
procedure formulate_content_id
(i_file_id in number,
 i_entity_name in varchar2,
 i_pk1_value in varchar2,
 i_pk2_value in varchar2,
 i_pk3_value in varchar2,
 i_pk4_value in varchar2,
 i_pk5_value in varchar2,
 x_cid out varchar2);
```

**Description**

Called by register\_attachment API to determine the unique CID for an attachment deposited into FND. This API is exposed for use outside of the register\_attachment API context.

**Arguments (input)****i\_file\_id**

File identifier from the FND\_LOBS table used to determine the file\_name value from the FND\_LOBS table. The file\_name is used to compose the x\_cid value.

**i\_entity\_name**

Entity name from the FND\_LOBS table used to compose the x\_cid value.

**i\_pk1\_value**

Key value from the FND\_LOBS table used to compose the x\_cid value.

**i\_pk2\_value**

Key value from the FND\_LOBS table used to compose the x\_cid value.

**i\_pk3\_value**

Key value from the FND\_LOBS table used to compose the x\_cid value.

**i\_pk4\_value**

Key value from the FND\_LOBS table used to compose the x\_cid value.

**i\_pk5\_value**

Key value from the FND\_LOBS table used to compose the x\_cid value.

#### Arguments (output)

##### **x\_cid**

A unique identifier for the attachment based on the concatenation of i\_entity\_name, i\_pk1\_value, i\_pk2\_value, i\_pk3\_value, i\_pk4\_value, i\_pk5\_value (which are provided as input to this API), and then adding the file\_name value (based on the file\_id) from the FND\_LOBS table.

## APIs Defined in ECX\_ENG\_UTILS

### **convert\_to\_cxml\_date**

#### PL/SQL Syntax

```
procedure convert_to_cxml_date
(p_ora_date in date,
 x_cxml_date out varchar2);
```

#### Description

Converts Oracle date to cXML date in ISO 8601 format. The time element is not included. If both date and time are required, use convert\_to\_cxml\_datetime procedure.

#### Arguments (input)

##### **p\_ora\_date**

The Oracle date.

#### Arguments (output)

##### **x\_cxml\_date**

The cXML date converted from the Oracle date.

### **convert\_to\_cxml\_datetime**

#### PL/SQL Syntax

```
procedure convert_to_cxml_datetime
(p_ora_date in date,
 x_cxml_date out varchar2);
```

#### Description

Converts Oracle date and time to cXML date and time in ISO 8601 format.

If only the date is required, use convert\_to\_cxml\_date procedure.

**Important:** Be sure to set the profile option ECX: Server Time Zone. If you do not set this profile option, the time zone defaults to GMT. See Define System Profile Options, page 3-2.

#### Arguments (input)

##### **p\_ora\_date**

The Oracle date and time.

#### Arguments (output)

##### **x\_cxml\_date**

The cXML date and time converted from the Oracle date and time.

#### **convert\_from\_cxml\_datetime**

#### PL/SQL Syntax

```
procedure convert_from_cxml_datetime
(p_cxml date in varchar2,
 x_ora_date out date);
```

#### Description

Converts cXML date and time to Oracle date and time.

**Important:** Be sure to set the profile option ECX: Server Time Zone. If you do not set this profile option, the time zone defaults to GMT. See Define System Profile Options, page 3-2.

#### Arguments (input)

##### **p\_cxml\_date**

The cXML date and time on an incoming document.

#### Arguments (output)

##### **x\_ora\_date**

The Oracle date and time converted from the cXML date and time.

---

# Troubleshooting

## Troubleshooting Your XML Gateway Installation

This section presents the following troubleshooting topics:

- Automated Troubleshooting Script, page G-1
- Transaction Monitor, page G-5
- Manual Troubleshooting Steps, page G-11
- XML Gateway Version Validation, page G-31
- Common Client Authentication Implementation Issues, page G-33
- Oracle Diagnostic Tests, page G-34

For additional troubleshooting information, see "Oracle XML Gateway Troubleshooting Guide" on *OracleMetaLink*.

### Automated Troubleshooting Script

Use the `ecxver.sql` script to verify the health of your Oracle XML Gateway installation. It can be run at any time. Outlined below are instructions for executing the script and interpreting the output.

#### How to Run the Automated Test Script

Log in to your database applications account and execute the following:

```
SQL> @$ECX_TOP/patch/115/sql/ecxver.sql
```

The output of the script will appear on your screen.

## Sample Output of the Automated Test Script

Component	OBJECT_NAME	LOCKED_MODE
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
OTA	ECX_OUTQUEUE	Normal
WEBMETHODS	ECX_OUTQUEUE	Normal
WEBMETHODS	ECX_OUTQUEUE	Normal
WEBMETHODS	ECX_OUTQUEUE	Normal
WEBMETHODS	ECX_OUTQUEUE	Normal
WEBMETHODS	ECX_OUTQUEUE	Normal

```

ECX_UTL_XSLT_DIR Profile : /sqlcom/log/ecx115
ECX_OAG_LOGICALID Profile : www.oracle.com
ECX_SERVER_TIMEZONE Profile:
ECX_SYS_ADMIN_EMAIL Profile: fname_lname@oracle.com
ECX_UTL_LOG_DIR Profile : /sqlcom/log/ecx115
ECX_XML_VALIDATE_FLAG Profile: Y
ECX_XML_MAXIMUM_SIZE Profile: 2000000
utl_file_dir: /sqlcom/log/ecx115
Oracle XML Parser: 2.0.2.9.0    Production
Parser Version OK

```

```

-----
XML Gateway Status Summary
-----

```



---

Log Profile/utl_file_dir	OK
XML Parser Version	OK
All ECX Objects Valid?	OK
All XML Parser Objects Valid?	OK
webMethods Running?	OK
OTA Running?	OK
Total Messages on Outbound Queue	6
OTA Msgs on Outbound Queue	4
webMethods Msgs on Outbound Queue	1
Messages on Inbound Queue	8

---

-----  
End of Summary  
-----

### How to Interpret the Output Generated by the Automated Test Script

The following table explains each line of the example output shown above. Some of the output is informational only and does not require further action.

Output	Explanation/Troubleshooting Tip
Transport Software	The "LOCKED_MODE" value should be "Normal", but may appear as "DEADLOCK". Deadlocks are normal if they remain for a short period of time. If a deadlock persists for more than 10 minutes, there may be a serious problem. <b>Note:</b> webMethods is used in Oracle Exchange environments only. It is not used in ERP environments
System Profiles: ECX_UTL_XSLT_DIR ECX_OAG_LOGICALID ECX_SERVER_TIMEZONE ECX_SYS_ADMIN_EMAIL ECX_UTL_LOG_DIR ECX_XML_MAXIMUM_SIZE ECX_XML_VALIDATE_FLAG	The value associated with each XML Gateway System Profile is displayed for review.
ERROR: Parser Version Wrong	The XML Parser version associated with the environment is displayed. XML Parser version 2.0.2.9 or higher is required.
ERROR: Some Invalid ECX Objects	Informational message indicating some invalid ECX database objects were identified.
Log Profile/utl_file_dir	If value displayed is anything other than "OK", verify the following: The physical directories associated with the ECX_UTL_XSLT_DIR and ECX_UTL_LOG_DIR system profiles must be assigned to the utl_file_dir parameter in the INIT.ORA file. Also, make sure the physical directories are write-enabled. If necessary, make the changes and bounce the database.
XML Parser Version	If value displayed is "FAIL", upgrade XML Parser to version 2.0.2.9 or higher.
All ECX Objects Valid?	If value displayed is "NO", the invalid objects must be recompiled. Recompile the invalid objects or reapply the XML Gateway patch to resolve the invalid ECX database objects.
All XML Parser Objects Valid?	If value displayed is "NO", the XML Parser must be reinstalled. Reinstall the XML Parser to resolve the invalid XML Parser objects.
webMethods Running?	If value displayed is anything other than "OK", restart webMethods. For an Oracle-hosted environment: contact Operations For self-hosted environments: contact webMethods <b>Note:</b> webMethods is used in Oracle Exchange environments only. It is not used in ERP environments.
OTA Running?	If value displayed is anything other than "OK", restart Oracle Transport Agent (OTA). Follow instructions in OTA patch to start the Apache web server. Ensure the required JSERV properties are set.
Total Messages on Outbound Queue	Number reported represents the number of messages on the outbound queue awaiting processing by OTA or webMethods. If number is unusually high, check the individual totals in the OTA and webMethods queues.
OTA Messages on Outbound Queue	Number reported represents the number of messages on the outbound queue awaiting processing by OTA. This is a subset of the total. If number is unusually high, ensure OTA is running.
webMethods Messages on Outbound Queue	Number reported represents the number of messages on the outbound queue awaiting processing by webMethods. This is a subset of the total. If number is unusually high, ensure webMethods is running.
Messages on Inbound Queue	Number of messages on inbound queue (informational). As long as the agent for the inbound queue is enabled, the agent listener is running, and OTA/webMethods is running, these messages will be processed. See Start Agent Listeners, page 6-52 for information on enabling agents and starting agent listeners.

## Transaction Monitor

The Transaction Monitor is a tool for monitoring the status of inbound and outbound transactions originating from and going into the Oracle e-Business Suite that have been processed by the XML Gateway and delivered or received by the Oracle Transport Agent. The Transaction Monitor shows a complete history and audit trail of these documents. You can also use the Transaction Monitor to resend an outbound document, if necessary.

Navigate to the Transaction Monitor page using the Workflow Administrator Web (New) responsibility.

The Transaction Monitor provides the following:

- Flexible search criteria to support access to a specific document or group of documents
- Search results at the document header level with drill down by document ID
- Resend capability for outbound messages
- Viewing capability of the XML message content

## Transaction Monitor Search Page

Use the Transaction Monitor Search page to enter search criteria for a specific inbound or outbound document, or group of documents.

The LOVs contain the valid search values and are defined under the Lookup Type FND\_TX\_MONITOR\_STATUS. The search page allows you to search on the following criteria:

### Inbound Messages

Select this radio button to search for inbound messages, and then select a Processing Status.

#### Processing Status

Select the processing status of the inbound message(s) for which you want to search. Valid values are All, Pending, Warning, Error, and Success.

### Outbound Messages

Select this radio button to search for outbound messages, and then select the Generation Status, Delivery Status, and Retry Status.

#### Generation Status

Select the generation status of the outbound message(s) for which you want to search. Valid values are All, Pending, Warning, Error, and Success.

**Note:** A Generation Status of "Success" indicates that the message was generated and enqueued.

#### Delivery Status

Select the delivery status of the outbound message(s) for which you want to search. Valid values are All, Pending, Warning, Error, and Success.

**Note:** A Delivery Status of "Success" indicates that the message was delivered successfully.

#### **Retry Status**

Select the retry status of the outbound message(s) for which you want to search. Valid values are All, Pending, Warning, Error, and Success.

#### **Transaction Type**

The transaction type is the product short name for the base Oracle Application associated with the transaction. Select a value from the Search and Select window.

#### **Transaction Subtype**

The transaction subtype is a code relating to the transaction type. Select a value from the Search and Select window.

#### **Source TP Location Code**

The source TP location code is the source trading partner of the message. Select a value from the Search and Select window.

#### **Trading Partner Name**

Select the trading partner name from the Search and Select window.

#### **Document ID**

Enter the document ID provided by Oracle e-Business Suite.

#### **Site Name**

Select the site name associated with the Trading Partner from the Search and Select window.

#### **Party Type**

Select the party type from the LOV. To search for all party types, select the blank LOV option.

#### **From Date**

Select the from date using the calendar icon, or enter a date in the format dd-mm-yyyy:hh:mm:ss (Example: 23-10-2002).

#### **To Date**

Select the to date using the calendar icon, or enter a date in the format dd-mm-yyyy:hh:mm:ss (Example: 23-10-2002). The to date must be equal to or after the from date. If the to date is the same as the from date, you are specifying one 24-hour period.

#### **Calling the Transaction Monitor Search Page from other applications**

**Note:** The Transaction Monitor Search page is a callable URL from any environment. Call the page by using the following URL and parameters:

```
OA.jsp?OAFunc=TXMONITORRESULTSPG&addBreadCrumb=Y&Direction=
<valu
e_direction>&...(other parameter-value pairs)
```

<value\_direction> should be IN or OUT

addBreadCrumb=Y is added to maintain the breadcrumbs trail

Additional parameters that can be used to call the Transaction Monitor are:

ProcessingStatus (with Direction=IN)

GenerationStatus (with Direction=OUT)

DeliveryStatus (with Direction=OUT)

RetryStatus (with Direction=OUT)

PartyType (lookup values CARRIER, S, I, B, E, C)

TransactionType

TxSubtype

LocCode

TPName (must be submitted with PartyType)

DocumentId

SiteName (must be submitted with PartyType)

From (in format DD-MON-YYYY HH:MI:SS)

To (in format DD-MON-YYYY HH:MI:SS)

## Search Results Page

The Transaction Monitor will return the Inbound Search Results page or the Outbound Search Results page depending on your selection. The document ID drills down to a details page from which you can view the XML message. For outbound documents, the Search Results page allows you to resend a document.

## Inbound Search Results Page

### Document ID

The document ID of the document being processed, such as the PO number or the Invoice number. This field drills down to the Message Details page.

### Trading Partner Name

The trading partner name. A value of "N/A" indicates that the XML Gateway Trading Partner Details have not been synchronized with the Workflow Directory Services. For more information, see: Setting up an Oracle Workflow Directory Service in the *Oracle Workflow Administrator's Guide*.

## External Transaction Type

The external transaction type is the primary external identifier for the XML message (for example, OAG noun). It is associated with the internal transaction type and transaction subtype.

## External Transaction Subtype

The external transaction subtype is a code relating to the transaction subtype (for example, OAG verb). It is the secondary identifier for the XML message.

**Note:** The combination of Transaction Type, Transaction Subtype, and the External Transaction Type and External Transaction Subtype identifies an Oracle transaction with which to associate this message.

## Processing Time Stamp

The date and time the document was processed.

## Processing Status

The processing status of an inbound message. Valid values are Pending, Warning, Error, and Success.

## Outbound Search Results Page

The Outbound Search Results page allows you to resend a document. Select the document(s) and click the Resend button. This action will not recreate the document, but resends the document from information stored in the `ecx_doclogs` table.

**Note:** The Select option is disabled for transactions that are A2A (because no message ID is created), and for B2B transactions for which generation failed.

## Document ID

The document ID drills down to the message detail screen.

## Trading Partner Name

The trading partner name. A value of N/A indicates that the XML Gateway Trading Partner Details have not been synchronized with Workflow Directory Services.

## Internal Transaction Type

The transaction type is the product short name for the base Oracle Application associated with the transaction.

## Internal Transaction Subtype

The transaction subtype is a code relating to the transaction type.

## Generation Date

The date and time the message was generated.

## Generation Status

Possible values of the generation status are: Pending, Warning, Error, and Success.

**Note:** A Generation Status of "Success" indicates that the message was generated and enqueued.

**Delivery Date**

The date and time the message was delivered.

**Delivery Status**

Possible values of delivery status are: Pending, Warning, Error, and Success.

**Note:** A Delivery Status of "Success" indicates that the message was delivered successfully.

**Retry Date**

The "retry" date of the message. The Retry Status column is updated if Resend was initiated from the search detail window.

**Retry Status**

Possible values are: Pending, Warning, Error, and Success.

**Resend (button)**

Use this button to resend selected documents from information stored in the ecx\_doclogs table. This function will not recreate the document. You can resend a single document, or select multiple documents to resend.

**Transaction Monitor Details Page**

The details screen is displayed when you drill down on the Document ID field.

**Inbound Message Details**

The Inbound Message Details Screen displays information regarding the selected document. Use the View XML button to view the XML document.

Fields displayed that are not shown on the Inbound Search Results screen, page G-7 are:

**Internal Control Number**

The unique number generated by the execution engine.

**Party ID**

The Trading Partner identifier.

**Party Type**

The type of Party, such as Customer, Supplier, or Bank.

**Site Name**

The trading partner's site name.

**Processing Message**

A message regarding the status of the transaction. Refer to Manual Troubleshooting, page G-11 for information on how to interpret messages and resolve reported errors.

**Processing Logfile**

The name and location of the processing log file. Use it to trace the process flow and identify errors.

**View XML (button)**

Clicking this button displays the XML message in the View XML page, page G-11.

**Outbound Message Details**

Click on the Document ID to drill down to the Transaction Monitor Details page.

Fields displayed that are not shown on the Outbound Search Results screen, page G-8 are:

**Party ID**

The identifier of the Trading Partner you are doing business with.

**Site Name**

The site name associated with the Trading Partner.

**URL Sent To**

The URL the document was sent to.

**Generation Status**

Valid values are Success, Error, Pending, or Warning.

**Generation Message**

The message returned by the system regarding the status of the transaction. For information regarding the interpretation of this message, see Manual Troubleshooting Steps, page G-11.

**Generation Logfile**

The name and location of the logfile. If this is a failed transaction, see Manual Troubleshooting Steps, page G-11.

**Delivery Message**

The message returned by the system regarding the status of the delivery.

**Retry Message**

The message returned by the system regarding the status of the retry message.

**View XML (button)**

Select this button to display the XML message in the View XML, page G-11 page. This button is enabled when the Generation Status is "Success".



## View XML

This page displays the XML document.

**Note:** The information in the "View XML" screen sample shown below was taken from the `ecx_doclogs` table on the basis of the out message ID stored in the view.

## Manual Troubleshooting Steps

If this is a new install or a new upgrade and you are experiencing errors, run the verification script included in the XML Gateway patch to verify that all the necessary components of the XML Gateway solution are installed correctly. Refer to Automated Troubleshooting Script, page G-1 for details on how to execute the troubleshooting script, `ecxver.sql`.

If the troubleshooting script did not reveal any install or upgrade errors, or if the error occurs in the processing stage, perform the following manual troubleshooting procedures.

There are three types of error messages generated by Oracle XML Gateway:

- Engine-level, page G-11 - error messages generated by the XML Gateway execution engine.
- API-level, page G-26 - error messages generated by APIs.
- HTTP Response Codes, page 7-22 - error response codes returned in the HTTP Response by the OTA server

## XML Gateway Engine-Level Messages

The following is a list of the seeded error messages that may be included in a notification to the Trading Partner or System Administrator. The same error message may also appear in the process log file depending on the trace level activated. Along with each error message is an explanation of possible causes and recommended corrective actions.

The list is organized by functional area and error type for easy reference. The list contains the following columns:

- Error Type
- Error Code
- Message Code
- Message
- Corrective Action

The Error Type identifies the target recipient of a notification. The target recipient is predetermined and is based on which party is most able to resolve the error. In a few instances, a notification is sent to both the Trading Partner and System Administrator so that the two parties may collaborate to resolve the error. The valid Error Types are listed in the following table:

Error Type	Description
10	No notification
20	Notification sent to Trading Partner
25	Notification sent to both Trading Partner and System Administrator
30	Notification sent to System Administrator

The Error Code identifies the status of the process. The valid Error Codes are shown in the following table:

Error Code	Description
10	In Process
0	Success
1	Warning
2	Error

Because the focus of this section is on troubleshooting, Error Type 10 (No Notification) and Error Code 0 (Success) are excluded from the list. Error Code 10 (In Process) is included to support troubleshooting, although this group does not represent an error. The engine will abort and roll back any process that does not have a status of "Success."

The Message Code is a code related to the message text string. This is used for quick identification of an error.

There are several error messages that can originate from multiple sources. The Message Code and Message string may be the same, but the Error Type and Error Code will be different. Ensure not to regard these as duplicates.

There are several categories of corrective action identified in the troubleshooting guide. The following table summarizes the categories and lists the corresponding reference section in this book:

Corrective Action Category	Reference Section
Verify lookup data	XML Gateway Setup chapter: Define Lookup Values, page 3-12
Verify development setup data	XML Gateway Setup chapter: Define XML Standards, page 3-6 Define Transactions, page 3-7
Verify implementation setup data	XML Gateway Setup chapter: Define System Profile Options, page 3-2 Define Trading Partner, page 3-13 Define Code Conversion, page 3-34
Verify queues and agent listeners are enabled	Integrating XML Gateway with Workflow Business Event System chapter: Manage Workflow Processes, page 6-49 Monitor Workflow Processes, page 6-51
Enable trace or configure event subscriptions	Integrating XML Gateway with Workflow Business Event System chapter: Manage Workflow Processes, page 6-49 Monitor Workflow Processes, page 6-51
Modify a message map: Action Definition	Message Designer chapter (see note following table): Reference the specific action under Transaction Map - Actions, page 2-51
Modify a message map: Root Element	Message Designer chapter (see note following table): File > Properties menu option, page 2-4
Load/Delete a map or DTD	Message Designer chapter (see note following table): How to Load/Delete Message Maps and DTDs, page 2-87
Verify Confirmation Message setup	Execution Engine chapter: How to Implement a Confirmation Message, page 4-13
Enter a bug	Work with Oracle World Wide Support to enter a bug. Include log file, message map, (.xgm file), and the associated DTD in the bug report.
Enter an enhancement request	Work with Oracle World Wide Support to enter an enhancement request. Include business justification and your specific scenario. Be as specific as possible.

**Note:** Updated maps must be loaded into the XML Gateway repository. Changes in the relationship between a map (.xgm) and the corresponding DTD may require the map, the DTD, or both to be loaded into the XML Gateway repository.

### UTL\_FILE Errors:

Error Type	Error Code	Message Code	Message	Corrective Action
30	1	ECX_UTL_INVALID_OPERATION	UTL FILE Error: The file could not be opened or operated on as requested.	Verify that the file is read and write enabled. Use commands appropriate for your operating system to change protections if necessary.
30	1	ECX_UTL_INVALID_PATH	UTL FILE Error: File location or file name was invalid.	Verify that the directory identified by the profile option exists. Verify that the file name exists. Change the directory or file name if necessary.
30	1	ECX_UTL_WRITE_ERROR	UTL FILE Error: An operating system error occurred during the write operation.	Verify the setting for the directory identified by the profile option. Change UTL_FILE_DIR setting in INIT.ORA if necessary and bounce the database. Verify that the directory and file exist and are write-enabled. Use commands appropriate for your operating system to change protections if necessary. Change the directory or file name if necessary.

### Trading Partner Errors:

Error Type	Error Code	Error Message Code	Message	Corrective Action
25	1	ECX_NO_UNIQUE_TP_SETUP	Could not resolve a Unique/Destination Configuration for this Partner.	The inbound message envelope contained a Trading Partner ID that was not uniquely defined in XML Gateway. This occurs when the wrong ID was sent or the Trading Partner has not been defined in XML Gateway. Use the window to view/add the Trading Partner profile.
30	1	ECX_ADDR_DERIVATION_ERR	Unable to derive internal address ID for address type: &p_address_type and location code: &p_location_code.	Cannot successfully execute Derive Address ID from Location Code action. This occurs when the address type or the location code is invalid. Check the section of the Message Designer chapter for a list of supported address types. Verify that the location code is valid.

Error Type	Error Code	Error Message Code	Message	Corrective Action
30	1	ECX_DELIVERY_TP_NOT_SETUP	Unable to determine the trading partner setup for Transaction type: &p_transaction_type, Transaction subtype: &p_transaction_subtype, Party site id: &p_party_site_id, Party id: &p_party_id, and Party type: &p_party_type.	Cannot determine Trading Partner with token values for Party Type, Party ID, Party Site ID, Transaction Type, and Transaction Subtype. Use the window/API to view/update the Trading Partner setup data.
30	1	ECX_DESTINATION_ADDR_NULL	Protocol Address missing for party id: &p_party_site_id and location code: &p_source_code.	The Protocol Address is null for the given Trading Partner (Party Site ID) and Source Location Code. Use the window to view/update the Trading Partner's protocol address.
30	1	ECX_DYN_ROUTING_NOT_ENABLED	&p_ext_type and &p_ext_subtype Not enabled for &p_party_ext_code in Oracle XML Gateway Server for Dynamic Routing.	The inbound transaction associated with the given external transaction type, transaction subtype, and source location code is not enabled for dynamic routing. Ensure that the "attribute3" variable in the message envelope contains a valid value. Use the window to view/update the Trading Partner Routing information representing the route to the Trading Partner.
30	1	ECX_INVALID_PARTY_TYPE	Party Type: &p_party_type is invalid.	Review seeded Party Type lookup and use a valid party type code. Enter an Enhancement Request for new Party Type if required.
30	1	ECX_INVALID_TP_HDR_ID	Tp_header_id: &p_tp_header_id is invalid.	Invalid Trading Partner ID. Use the retrieve/update APIs in ECXTPXFB to query/update a Trading Partner profile.
30	1	ECX_INVALID_TXN_PARAMS	Party Site ID or Transaction Type or Transaction Subtype cannot be NULL.	Use the window to view/update the Trading Partner profile and ensure that the Trading Partner Site, Transaction Type, and Transaction Subtype are valid.

Error Type	Error Code	Error Message Code	Message	Corrective Action
30	1	ECX_NO_EMAIL_ADDR	Unable to determine TP email address for party_type: &p_party_type, transaction_type: &p_transaction_type, transaction_subtype: &p_transaction_subtype, and party_site_id: &p_party_site_id.	Cannot determine Trading Partner e-mail address with given token values to send notification. Use the window to review/update the e-mail address in the Trading Partner setup.
30	1	ECX_NO_ENVELOPE	Unable to determine envelope information for internal control number: &p_icn.	Envelope information for the given internal control number was not found. The internal control number is a system-generated number that uniquely identifies the XML message being processed. Verify that the internal control number is valid.
30	1	ECX_NO_UNIQUE_TP_SETUP	Could not resolve a Unique/Destination Configuration for this Partner.	Cannot determine unique Trading Partner detail for outbound transaction with the information identified. Check parameters for Party Type, Party ID, Party Site ID, Transaction Type, and Transaction Subtype.
30	1	ECX_PARTY_ID_NOT_NULL	Party ID is a required parameter.	Pass valid Party ID, Party ID cannot be null.
30	1	ECX_PARTY_SITE_ID_NOT_NULL	Party Site ID is a required parameter.	Pass valid Party Site ID, Party Site ID cannot be null.
30	1	ECX_PARTY_TYPE_NOT_NULL	Party Type is a required parameter.	Review seeded Party Type lookup and pass valid Party Type. Party Type cannot be NULL.
30	1	ECX_PARTY_TYPE_NOT_SET	Please provide party_type for the trading partner.	Party Type not available for XML Gateway to uniquely determine Trading Partner. Provide valid Party Type from seeded Party Type lookup.
30	2	ECX_PARTY_TYPE_NOT_SET	Please provide party_type for the trading partner.	Party type not available for XML Gateway to uniquely determine Trading Partner. Provide valid Party Type from seeded Party Type lookup.

Error Type	Error Code	Error Message Code	Message	Corrective Action
30	1	ECX_RCVR_NOT_SETUP	Receiver TP setup not found for tp_header_id: &p_tp_header_id.	The Receiving Trading Partner was not found. Use the window to verify that the Trading Partner is defined.
30	2	ECX_RULE_INVALID_TP_SETUP	The Standard: &p_standard_code, Transaction Type: &p_transaction_type, Transaction Subtype: &p_transaction_subtype, and Location Code: &p_party_site_id is not enabled in the XML Gateway Server. Please check your Setup.	Either invalid transaction, standard code, or Trading Partner encountered. Use the Define Transactions window to view/update the transaction setup. Use the window/API to view/update the Trading Partner setup and the transactions enabled for the the Trading Partner.
30	1	ECX_SNDR_NOT_SETUP	Sender TP setup not found for tp_header_id: &p_tp_header_id.	The Sending Trading Partner was not found. Use the window to verify that the Trading Partner is defined.
30	1	ECX_STATIC_ROUTING_NOT_ENABLED	&p_ext_type and &p_ext_subtype Not enabled for &p_party_ext_code in Oracle XML Gateway Server for Static Routing.	Could not determine routing information for given Trading Partner and transaction. Use the window to view/update the Trading Partner details for the Routing ID.
30	1	ECX_TRANSACTION_NOT_ENABLED	&p_ext_type and &p_ext_subtype Not enabled for &p_party_ext_code in Oracle XML Gateway Server.	Inbound transaction not enabled in XML Gateway. Check envelope parameter values for Message Standard, Transaction Type, Transaction Subtype, and Source Trading Partner Location. Either the envelope information contains an error or the Trading Partner has not been defined. Use the window to define the Trading Partner to process this inbound transaction.
30	1	ECX_TRANSACTION_NOT_FOUND	No Transaction defined for Transaction Type: &p_transaction_type, Transaction Subtype: &p_transaction_subtype, and Party Type: &p_party_type.	Given Party Type, Transaction Type, and Transaction Subtype not found. Check parameter values being passed.

Error Type	Error Code	Error Message Code	Message	Corrective Action
30	2	ECX_USER_TP_VALID	The Standard: &p_standard_code, Transaction Type: &p_transaction_type, Transaction SubType: &p_transaction_subtype, Location Code: &p_party_site_id and User Name:&p_user_name is not enabled in the XML Gateway Server. Please check your Setup.	Either invalid standard code, transaction, user name, or Trading Partner encountered. Use the Trading Partner User Setup form to associate a user with a specific Trading Partner. Use the Trading Partner Setup form to view or update the Trading Partner setup.
30	2	ECX_TP_USER_ASSIGNED	The user &user_name is already assigned to another Trading Partner.	Assign different users to the Trading Partner in the Trading Partner User Setup form.
30	2	ECX_TP_INSUFFICIENT_VAL	Please save the header record before selecting User Setup.	In the Trading Partner Setup form, save the Trading Partner header record including Trading Partner Type, Trading Partner Name, Trading Partner Site, and Company Admin Email first before clicking the User Setup button to have access to the Trading Partner User Setup form.

### Hub Errors

Error Type	Error Code	Error Message Code	Message	Corrective Action
25	1	ECX_DELIVERY_HUB_NOT_SETUP	Unable to resolve the hub destination parameters for this trading partner hub.	Cannot determine the hub attributes (for inbound transaction) for this Trading Partner. Use the Define Hubs and windows to view/update hub attributes. Hub attributes are used if Trading Partner "Connection/Hub" attribute is set to "Hub."
30	1	ECX_DELIVERY_HUB_NOT_SETUP	Unable to resolve the hub destination parameters for this trading partner hub.	Cannot determine the hub attributes (for outbound transaction) for this Trading Partner. Use the Define Hubs and windows to view/update hub attributes. Hub attributes are used if Trading Partner "Connection/Hub" attribute is set to "Hub."



## Code Conversion Errors

Error Type	Error Code	Error Message Code	Message	Corrective Action
30	1	ECX_CODE_CONVERSION_DISABLED	Standard Code: &MESSAGE_STANDARD not Found. Code Conversion is disabled.	The execution engine is using an XML standard not defined in ECX_STANDARDS, so the code conversion cannot be performed. Use the Define XML Standards window to review the seeded XML standards. Enter an enhancement request for required XML standard and identify the transactions that require this new standard.
30	1	ECX_TP_NOT_FOUND	Trading Partner not Found. Code Conversion is disabled.	A warning indicating that the Trading Partner was not found, so Trading Partner code conversion cannot be performed. Use the window to view/update the Trading Partner setup.

## Enqueue/Dequeue Errors

Error Type	Error Code	Error Message Code	Message	Corrective Action
10	10	ECX_DEQUEUED_LOGGED	Message dequeued and logged	Message dequeued and logged.
30	2	ECX_MANY_PROCESSING_QUEUES	More than one row resulted while querying the queue name.	Unable to determine unique queue for inbound transaction. Use the Define Transactions window to view/update the queue definition for the inbound transaction being processed.
30	1	ECX_NO_PROCESSING_QUEUE	Unable to determine processing engine queue.	Unable to determine processing queue for inbound transaction. Use the Define Transactions window to view/update the queue definition for the inbound transaction being processed.
30	1	ECX_PROCESSING_ENQ_ERROR	Error enqueueing to processing engine: &p_queue_name	Unable to enqueue when processing inbound transaction. Ensure that the transaction queue and agent listener are enabled.
30	1	ECX_REPROCESSING_ERROR	Error enqueueing to processing engine, while REPROCESSING: &p_out_queue	Unable to enqueue when reprocessing inbound transaction. Ensure that the transaction queue and agent listener are enabled.

## Workflow Business Event System Errors

Error Type	Error Code	Error Message Code	Message	Corrective Action
10	10	ECX_PROCESSING_RULE	Processing rule	Processing rule
30	2	ECX_BUSINESS_EVT_NOT_SET	Business Event Not Set for the Transaction	Check that the message map for the inbound transaction concludes with root level, postprocess procedure call action to call the ECX_STANDARD.setEventDetails API. This API sets the business event to be raised to indicate an inbound transaction has been successfully processed. The corresponding application event subscription will consume this event.
30	2	ECX_IN_RULE_PROCESSING_ERROR	Error in processing inbound rule	Unexpected exception when processing inbound rule. Check the setup for event and agent listener.

### Confirmation BoD Errors

Error Type	Error Code	Error Message Code	Message	Corrective Action
30	2	ECX_CONFIRM_DOC_NOT_FOUND	Unable to determine the inbound document for which this confirmation is being sent.	Verify that the Workflow for the outbound confirmation includes the internal control number corresponding to the inbound business document. Refer to the Workflow Development Guidelines on how to implement the Confirmation BoD for the details.
30	2	ECX_CONFIRM_GENERATE_FAILED	Cannot generate confirmation for Confirmation Inbound. Loop Detected.	Cannot use a Confirmation message to acknowledge a Confirmation message.
30	2	ECX_CONFIRM_STATUS_NOT_FOUND	Cannot find confirmation status for the trading partner.	Could not determine the Confirmation flag setting for the Trading Partner. Use the window to view/update the Trading Partner setup and set the Confirmation flag if necessary.
30	2	ECX_CONFIRM_TP_NOT_SETUP	Unable to determine trading partner setup for the Confirmation. Transaction Type: &p_transaction_type, Transaction Subtype: &p_transaction_subtype, Location Code: &p_location_code.	Cannot determine Trading Partner to send the Confirmation. Use the window to view/update the Trading Partner setup for the given Transaction Type, Transaction Subtype, and Source Trading Partner Location Code.

### Message Map and DTD Errors

Error Type	Error Code	Error Message Code	Message	Corrective Action
20	1	ECX_INCOMPLETE_OAG_DATE	Incomplete OAG Date: &p_datetime.	A warning indicating the incoming OAG date was incomplete. This could cause the conversion to the Oracle date to be incomplete.
20	1	ECX_USER_INVOKED_EXIT	Program Exit invoked by the User.	A warning indicating a user-programmed exit was executed.
30	1	ECX_CANNOT_CONVERT_TO_DATE	Cannot Convert to Date.	Usage of (or other data conversion APIs) in the message map triggered an error. Use Message Designer to verify that the map action/API is defined correctly. Make the necessary changes and reload the updated map.

Error Type	Error Code	Error Message Code	Message	Corrective Action
30	2	ECX_CANNOT_CONVERT_TO_NUM	Cannot convert the value &p_value to number.	Invalid number used with math function. Verify that the variable/literal value is numeric. Make the necessary changes and reload the updated map.
30	1	ECX_CANNOT_CONVERT_TO_NUMBER	Cannot Convert to number.	Condition evaluation error, inform the System Administrator or log a bug.
30	1	ECX_COND_NOT_DEFINED	Condition type: &TYPE not defined.	Review map actions using Message Designer to ensure that the condition expressions are defined correctly with two operands and a valid operator. Make the necessary changes and reload the updated map.
30	2	ECX_DATATYPE_CONV_FAILED	DataType conversion failed while inserting the level into the table.	Datatype mismatch between source and database datatype. Make the necessary changes and reload the updated map.
30	1	ECX_INVALID_ADDRESS_TYPE	Address type: &p_address_type is invalid.	Invalid address type. Check section of Message Designer chapter for list of supported address types.
30	1	ECX_INVALID_NUMBER	Binding value &pre_var_value for bind variable &variable_name is an invalid number.	Usage of the action in the message map contained an invalid number for the given Bind Value and Bind Variable combination. Use Message Designer to review the Source Definition of the map for all usage of the Append Where Clause, make the necessary changes, and reload the updated map.
30	1	ECX_MAPPINGS_NOT_FOUND	Mappings for Map ID: &MAP_ID not found.	Map details for given Map ID not found because incorrect mappings, such as level cross overs, were detected. Use Message Designer to review the message map. Make the necessary changes, and reload the updated map.
30	1	ECX_MATH_FUNC_NOT_NULL	Math function type cannot be null.	Usage of Math function is incomplete. Use Message Designer to review all math functions defined, make the necessary changes, and reload the updated map.

Error Type	Error Code	Error Message Code	Message	Corrective Action
30	1	ECX_ROOT_ELEMENT_NOT_FOUND	Root Element information not found for Map ID: &MAP_ID.	Root element information not found for given message map. Use the Message Designer to open the map and use the option to add the root element. Reload the updated map.
30	1	ECX_ROOT_INFO_NOT_FOUND	Map root element or parent node ID not found.	Root element or parent node ID not found for given message map. Use Message Designer to open the map, and use the option to add the root element or parent node ID. Reload the updated map.
30	1	ECX_SEED_DATA_NOT_FOUND	Seed Data is missing for Map ID: &MAP_ID.	Map data missing for given Map ID. Make sure you are operating in the desired database instance.
30	2	ECX_STACKVAR_NOT_FOUND	Stack Variable not found.	Error in processing global variables. Inform System Administrator or log a bug.
30	1	ECX_UNSUPPORTED_COND_TYPE	Unsupported condition type: &TYPE.	Review map actions using Message Designer to ensure that the condition expressions are defined correctly with two operands and a valid operator. Make the necessary changes and reload the updated map.
30	2	ECX_UNSUPPORTED_DATATYPE	Unsupported Data Type.	The message map contained usage of unsupported data types. Use the Message Designer to review the message map to verify that only the VARCHAR2, NUMBER, DATE, CHAR, and CLOB data types are used. Make the necessary map changes and reload the updated map.
30	1	ECX_UNSUPPORTED_DATE_COND	Unsupported Condition for Date.	Cannot use Condition Expression to compare dates. Use database functions instead to compare dates.
30	1	ECX_UNSUPPORTED_MATH_FUNC	Unsupported Math function.	XML Gateway supports the four basic math functions: add, subtract, multiply, and divide. Use database functions for other math functions.

Error Type	Error Code	Error Message Code	Message	Corrective Action
30	1	ECX_UNSUPPORTED_NUMBER_COND	Unsupported Condition for Number.	Unsupported condition for numbers. Verify that the condition operator is relevant for numbers. Make the necessary map changes and reload the updated map.
30	1	ECX_UNSUPPORTED_STRING_COND	Unsupported Condition for Strings.	Unsupported condition for strings. Verify that condition operator is relevant for strings. Make the necessary map changes and reload the updated map.

### Inbound Processing Errors

Error Type	Error Code	Error Message Code	Message	Corrective Action
10	10	ECX_REPROCESSING_MESSAGE	Reprocessing message.	Reprocessing message.
25	1	ECX_PROTOCOL_ADDR_NULL	Destination Address missing for party ID: &p_tp_detail_id.	Inbound message envelope did not contain a protocol address for the given Trading Partner. Use the window to view/update the Trading Partner's protocol address.
30	2	ECX_MSG_EXISTS_IN_LOG	Message already in the Log &p_msgid.	Inbound message already exists; duplicate not allowed.
30	1	ECX_MSGID_NOT_FOUND	Msg Id: &p_msgid not Found.	Inbound message for given message ID not found. Make sure the message ID is valid.
30	1	ECX_UNSUPPORTED_STANDARD	Not a Supported Standard in the XML Gateway Server.	Inbound message envelope indicates message is for an XML standard not supported by Oracle Applications. Use the Define XML Standards window to view the list of supported standards. Enter an enhancement request for the required XML standard and identify the transactions that require this new standard.

### General Processing Errors

Error Type	Error Code	Error Message Code	Message	Corrective Action
10	10	ECX_PROCESSING_MESSAGE	Processing message.	N/A
10	10	ECX_TRIGGER_OUTBOUND	Triggering outbound.	N/A

Error Type	Error Code	Error Message Code	Message	Corrective Action
25	1	ECX_PARSE_ERROR	Parse Error.	Parse error encountered because the generated or received XML message was poorly formed or invalid as per the DTD. The other possible reason for a parse error is that the DTD referenced in the map is not the same DTD that was passed. Make sure that the DTD referenced in the map is loaded into the XML Gateway repository.
30	1	ECX_DOCLOGS_NOT_EXISTS	No document log exists for message ID: &p_msgid.	ECX_DOCLOGS entry does not exist for the given message ID. Ensure that the message ID is valid.
30	1	ECX_DTD_NOT_FOUND	DTD not found for Map ID: &MAP_ID.	The DTD referenced in the message map was not found because the map referenced the wrong DTD or the DTD was not loaded into the XML Gateway repository. Use Message Designer to check the DTD referenced in the map, make the necessary changes, and reload the updated map. If necessary load the correct DTD into the XML Gateway repository.
30	2	ECX_ERROR_NOT_SET	Unable to set error information.	Inform System Administrator or log a bug.
30	1	ECX_MAP_NOT_FOUND	Not a Supported Map &p_map_code in the XML Gateway Server.	Given map code associated with the outbound transaction enabled for a Trading Partner is not supported. Use the window to view/update the map code associated with the enabled outbound transaction. Make sure the map has been loaded in the XML Gateway repository.
30	2	ECX_MAP_NOT_FOUND	Not a Supported Map &p_map_code in the XML Gateway Server.	Given map code associated with the inbound transaction enabled for a Trading Partner is not supported. Use the window to view/update the map code associated with the enabled inbound transaction. Make sure the map has been loaded in the XML Gateway repository.
30	1	ECX_MSGID_NOT_FOUND	Message ID cannot be NULL.	Pass valid value for Message ID parameter.
30	1	ECX_MSGID_NOT_NULL	Message ID cannot be NULL.	Pass valid value for Message ID parameter.

## XML Gateway API-Level Messages

The following is a list of messages generated by the XML Gateway APIs. The list is organized by API Return Code for easy reference. The list contains the following columns:

- API Return Code
- Message Code
- Message

The API Return Code represents the status of the API. The calling program can interrogate the return code and provide an application-specific error handler to address the error detected. Because these messages are designed for communication between application APIs, the necessary corrective actions are addressed during the development process. There is no action required, unless you are developing a custom transaction.

API Return Code	Description
0	Success
1	Warning
2	Unexpected Error
3	Null value detected for required parameter
4	Invalid parameter value
5	Cannot insert duplicate row
6	No data found for given parameter values
7	Duplicate row exists for given parameter values
8	Cannot delete due to referential integrity

Because the focus of this section is on troubleshooting, API Return Code of 0 (Success) is excluded from the following list. As of this printing, there are no seeded messages with API Return Code of 1 or 2.

API Return Code	Message Code	Message
3	ECX_CATEGORY_ID_NOT_NULL	Code category ID is a required parameter.
3	ECX_CODE_CATEGORY_NOT_NULL	Code category is a required parameter.
3	ECX_CONNECTION_TYPE_NOT_NULL	Connection type is a required parameter.
3	ECX_DATA_SEEDED_NOT_NULL	XRef data seeded is a required parameter.
3	ECX_DIRECTION_NOT_NULL	Direction is a required parameter.
3	ECX_EMAIL_ADDRESS_NOT_NULL	Email_address is a required parameter.
3	ECX_EXT_PROCESS_ID_NOT_NULL	External process ID is a required parameter.
3	ECX_HUB_USER_ID_NOT_NULL	Hub User ID is a required parameter.
3	ECX_LOCATION_NOT_NULL	Location Code is a required parameter.



API Return Code	Message Code	Message
3	ECX_MAP_CODE_NOT_NULL	Map Code is a required parameter.
3	ECX_PARTY_ID_NOT_NULL	Party ID is a required parameter.
3	ECX_PARTY_SITE_ID_NOT_NULL	Party Site ID is a required parameter.
3	ECX_PARTY_TYPE_NOT_NULL	Party Type is a required parameter.
3	ECX_PROTOCOL_TYPE_NOT_NULL	Protocol type is a required parameter.
3	ECX_PWD_NOT_NULL	For protocol type: &p_protocol_type, the password is a required parameter.
3	ECX_STANDARD_CODE_NOT_FOUND	Standard code: &p_standard_code not defined.
3	ECX_STANDARD_CODE_NULL	Standard Code is a required parameter.
3	ECX_STANDARD_NOT_NULL	Document standard is a required parameter.
3	ECX_TP_DTL_ID_NOT_NULL	Trading partner detail ID is a required parameter.
3	ECX_TP_HDR_ID_NOT_NULL	Trading partner header ID is a required parameter.
3	ECX_TRAN_SUBTYPE_NOT_NULL	Transaction Subtype is a required parameter.
3	ECX_TRANSACTION_SUBTYPE_NOT_NULL	Transaction type is a required parameter.
3	ECX_TRANSACTION_TYPE_NOT_NULL	Transaction type is a required parameter.
3	ECX_XREF_DTL_ID_NOT_NULL	XRef detail ID is a required parameter.
3	ECX_XREF_EXT_VAL_NOT_NULL	XRef external value is a required parameter.
3	ECX_XREF_INT_VAL_NOT_NULL	XRef internal value is a required parameter.
3	ECX_XREF_STANDARD_ID_NOT_NULL	XRef standard ID is a required parameter.
3	ECX_XREF_STD_VAL_NOT_NULL	XRef standard value is a required parameter.
3	ECX_USERNAME_NOT_NULL	For protocol type: &p_protocol_type, the username is a required parameter.
4	ECX_DATA_OWNER_INCONSISTENT	Data owner: &p_owner is inconsistent with data seeded flag: &p_data_seeded.
4	ECX_INVALID_CONF_CODE	Confirmation code: &p_confirmation is invalid.
4	ECX_INVALID_DIRECTION	Direction: &p_direction is invalid.
4	ECX_INVALID_EMAIL_ADDRESS	Email_address: &p_email_address is invalid.
4	ECX_INVALID_EXT_PROCESS_ID	External process ID: &p_ext_process_id is invalid.

API Return Code	Message Code	Message
4	ECX_INVALID_HUB_ID	Hub ID: &p_hub_id is invalid.
4	ECX_INVALID_HUB_USER_ID	Hub user ID: &p_hub_user_id is invalid.
4	ECX_INVALID_MAP_CODE	Map Code: &p_map_code is invalid.
4	ECX_INVALID_PARTY_ID	Party ID: &p_party_id is invalid.
4	ECX_INVALID_PARTY_SITE_ID	Party_site_id: &p_party_site_id is invalid.
4	ECX_INVALID_PARTY_TYPE	Party Type: &p_party_type is invalid.
4	ECX_INVALID_PROTOCOL_TYPE	Protocol type: &p_protocol_type is invalid.
4	ECX_INVALID_PWD	The password should not contain special characters.
4	ECX_INVALID_PWD_LEN	The length of the password should be greater than 5 characters.
4	ECX_INVALID_ROUTING_ID	Routing ID: &p_routing_id is invalid.
4	ECX_INVALID_TP_DETAIL_ID	Tp_detail_id: &p_tp_detail_id is invalid.
4	ECX_INVALID_TP_HDR_ID	Tp_header_id: &p_tp_header_id is invalid.
5	ECX_CODE_CATEGORY_EXISTS	Code category: &p_xref_category_code is already defined in the table.
5	ECX_DOCUMENT_STANDARD_EXISTS	Standard already exists with standard code &p_standard_code and standard type &p_standard_type.
5	ECX_DUPLICATE_TRANSACTIONS	Duplicate Rows found for Transaction Type: &p_transaction_type, Transaction Subtype: &p_transaction_subtype, and Party Type: &p_party_type.
5	ECX_TP_DTL_EXISTS	Trading Partner detail record for tp_header_id: &p_tp_header_id, ext_process_id: &p_ext_process_id is already defined in the table.
5	ECX_TP_DTL1_EXISTS	Trading Partner detail record for the combination of External Transaction Type: &p_ext_type, External Transaction Subtype: &p_ext_subtype, Document Standard ID: &p_standard_id, Source TP Location Code: &p_source_tp_location_code is already defined for the inbound transaction.
	ECX_TP_DTL2_EXISTS	Trading Partner detail already exists for this combination of tp_header_id: &p_tp_header_id, Transaction Type: &p_transaction_type, Transaction Subtype: &p_transaction_subtype for outbound transactions.

API Return Code	Message Code	Message
5	ECX_TP_HDR_EXISTS	Trading Partner for party_type: &p_party_type, party_id: &p_party_id, party_site_id: &p_party_site_id is already defined in the table.
5	ECX_XREF_DTL_ID_ROW_EXISTS	Duplicate code conversion record for header id: &p_tp_header_id, external value: &p_xref_ext_value, internal value: &p_xref_int_value, direction: &p_direction.
5	ECX_XREF_DTL_ROW_EXISTS	Duplicate code conversion record for code category: &p_xref_category_code, Standard code: &p_standard_code, external value: &p_xref_ext_value, internal value: &p_xref_int_value, direction: &p_direction.
5	ECX_XREF_STD_ID_ROW_EXISTS	Duplicate standard code conversion record for XRef standard ID: &p_xref_standard_id, Standard value: &p_xref_std_value, Internal value: &p_xref_int_value.
5	ECX_XREF_STD_ROW_EXISTS	Duplicate standard code conversion record for document standard: &p_standard, Standard value: &p_xref_std_value, Internal value: &p_xref_int_value.
6	ECX_CODE_CATEGORY_NOT_FOUND	Code category: &p_xref_category_code not defined.
6	ECX_HUB_NOT_EXISTS	No Hub definition exists for connection_type: &p_connection_type and protocol: &p_protocol_type.
6	ECX_NO_ROWS_DELETED	No rows deleted in table: &p_table, &p_param_name: &p_param_id is invalid.
6	ECX_NO_ROWS_UPDATED	No rows updated in table: &p_table, &p_param_name, &p_param_id is invalid.
6	ECX_NOT_FOUND	No rows fetched from table: &p_table and key: &p_key.
6	ECX_STANDARD_NOT_FOUND	Document standard: &p_standard not found.
6	ECX_STANDARD_ROW_NOT_FOUND	No standard found with standard code &p_standard_code and standard type &p_standard_type.
6	ECX_TP_DTL_NOT_FOUND	No trading partner detail records found for tp_header_id: &p_tp_header_id, ext_process_id: &p_ext_process_id.
6	ECX_TP_DTL1_NOT_FOUND	No trading partner detail records found for party_type: &p_party_type, party_id: &p_party_id, party_site_id: &p_party_site_id, transaction_type: &p_transaction_type, transaction_subtype: &p_transaction_subtype, standard code: &p_standard_code, direction.

API Return Code	Message Code	Message
6	ECX_TP_HDR_NOT_FOUND	No trading partner header records found for party type: &p_party_type, party_id: &p_party_id, party_site_id: &p_party_site_id.
6	ECX_TRANSACTION_NOT_FOUND	No Transaction defined for Transaction Type: &p_transaction_type, Transaction Subtype: &p_transaction_subtype, and Party Type: &p_party_type.
6	ECX_XREF_DTL_EXT_ID_NOT_FOUND	No code conversion detail records found for standard code: &p_standard_code, code category: &p_xref_category_code, tp header ID: &p_tp_header_id, external value: &p_xref_ext_value.
6	ECX_XREF_DTL_INT_ID_NOT_FOUND	No code conversion detail records found for standard code: &p_standard_code, code category: &p_xref_category_code, tp header ID: &p_tp_header_id, internal value: &p_xref_int_value.
6	ECX_XREF_DTL_NOT_FOUND	No code conversion detail records found for standard code: &p_standard_code, code category: &p_xref_category_code, direction: &p_direction, external value: &p_xref_ext_value, internal value: &p_xref_int_value.
7	ECX_STANDARD_TOO_MANY_ROWS	More than one row exists for the unique key-standard code: &p_standard_code and standard type: &p_standard_type.
7	ECX_TOO_MANY_ROWS	More than one row fetched from table: &p_table and key: &p_key.
7	ECX_TP_DTL_TOO_MANY_ROWS	Number of records fetched is greater than one for tp_header_id: &p_tp_header_id, ext_process_id: &p_ext_process_id.
7	ECX_TP_DTL1_TOO_MANY_ROWS	Number of records fetched is greater than one for party_type: &p_party_type, party_id: &p_party_id, party_site_id: &p_party_site_id, transaction_type: &p_transaction_type, transaction_subtype: &p_transaction_subtype, standard_code: &p_standard_code, direction
7	ECX_TP_HDR_TOO_MANY_ROWS	Number of records fetched is greater than one for party_type: &p_party_type, party_id: &p_party_id, party_site_id: &p_party_site_id.
7	ECX_TRANS_TOO_MANY_ROWS	Too many rows in transactions for Transaction Type: &p_transaction_type, Transaction Subtype: &p_transaction_subtype, and Party Type: &p_party_type.

API Return Code	Message Code	Message
7	ECX_XREF_DTL_EXT_TOO_MANY_ROWS	Number of records fetched is greater than one for standard code: &p_standard_code, code category: &p_xref_category_code, tp header ID: &p_tp_header_id, external value: &p_xref_ext_value.
7	ECX_XREF_DTL_INT_TOO_MANY_ROWS	Number of records fetched is greater than one for standard code: &p_standard_code, code category: &p_xref_category_code, tp header ID: &p_tp_header_id, internal value: &p_xref_int_value.
8	ECX_CODE_CATEGORY_REFERENCED	Unable to delete code category ID: &p_category_id because it is referenced in other tables.

## XML Gateway Version Validation

Associated with each data definition or map file are the major and minor version numbers indicating the version of Message Designer used to create the map. The numbers are available in the <ECX\_MAJOR\_VERSION> and <ECX\_MINOR\_VERSION> tags. Message Designer will compare the map version number to the Message Designer version number (available in the Help >About Menu) and will open the map only if the major versions are the same and the minor versions are the same or lower.

In addition, the map version numbers will be compared to the XML Gateway installed version number (available in WF\_RESOURCES.ECX\_VERSION) when you attempt to load the maps into the XML Gateway repository. The maps will be loaded only if the major versions are the same and the minor versions are the same or lower.

## Problem: Error while running LoadMap

**Version Incompatible: XML Gateway 2.10 map is not supported by currently installed XML Gateway 2.6.**

### Sample error message:

```
Java oracle.apps.ecx.loader.LoadMap apps apps $JDBC mymap.xgm Database Connect String
(PORT=1543)) (CONNECT_DATA=(SID=ecxcert))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=ap935sun)
)UserName apps
Connecting...
connected.
Installed XML Gateway Version: 2.6.0.0
Started Processing of the Document
Processing Instructions for xmlGateway=> MAP
Map Code: ECX_POO Using Old Map id: 19
Version Incompatible: XML Gateway 2.10 map is not supported by currently installed XML Gateway 2.6.0.0
```

### Solution: Upgrade installed XML Gateway

The following table shows the compatibility of other sample map and XML Gateway versions. The version numbers are available in the <ECX\_MAJOR\_VERSION> and <ECX\_MINOR\_VERSION> tags.

In the example version 2.6, 2 is the major version number, and 6 is the minor version number.

Installed XML Gateway Version	XGM/LOADER Version	Compatible/ Incompatible	Action
2.6.0.0	2.10.0.0	Incompatible	Upgrade XML Gateway
2.6.0.0	2.6.0.0	OK	None
2.10.0.0	2.6.0.0	OK	None
2.6.0.0	2.4.0.0	OK	None
2.6.0.0	1.1.0.0	Incompatible	Upgrade Map
2.6.0.0	3.1.0.0	Incompatible	Upgrade XML Gateway

### Problem: Error while processing transaction (using debug level 3)

#### Sample error message:

```
Enter ECX_UTILS.INITIALIZE
i_map_id==>19
Enter ECX_UTILS.CHECK_VERSION
i_major_version==>2
i_minor_version==>10
i_eng_version==>2.6.0.0
i_eng_major_version==>2
i_eng_minor_version==>6
Exit ECX_UTILS.CHECK_VERSION
[ECE_PROGRAM_ERROR] PROGRESS_LEVEL=ecx_utils.INITIALIZE
[ECE_ERROR_MESSAGE] ERROR_MESSAGE=Version Incompatible: XML Gateway
version 2.10 is not supported by currently installed XML Gateway
2.6.0.0
Exit ecx_utils.INITIALIZE
Clean-up i_stack, l_node_stack and i_tmpxmlExit
ECX_OUTBOUND.PROCESS_OUTBOUND_DOCUMENTS
ECX_UTILS.INITIALIZE: Version Incompatible: XML Gateway version 2.
10 is not supported by currently installed XML Gateway 2.6.0.0

Exit ECX_OUTBOUND.GETXML
```

### Solution: Upgrade installed XML Gateway

At runtime, the engine compares the version from the .xgm file to the installed XML Gateway version. The Loader and the Engine only support maps that have the same major version and same (or lower) minor version. XML Gateway will return an error message if the versions are incompatible.

The following table shows other sample error conditions while processing transaction:

Installed XML Gateway Version	XGM/LOADER Version	Compatible/ Incompatible	Action
2.6.0.0	2.10.0.0	Incompatible	Upgrade XML Gateway
2.6.0.0	2.6.0.0	OK	None
2.10.0.0	2.6.0.0	OK	None
2.6.0.0	2.4.0.0	OK	None
2.6.0.0	1.1.0.0	Incompatible	Upgrade Map
2.6.0.0	3.1.0.0	Incompatible	Upgrade XML Gateway

## Common Client Authentication Implementation Issues

The following are common errors encountered during client authentication.

For more information on these and other errors, see "Installing Oracle XML Gateway with Oracle Applications 11i," on [OracleMetaLink](#).

### **java.lang.NoSuchMethodError**

If you get this error, ensure that the wrapper.classpath parameters are set correctly in the jserv.properties file. See Setup Parameters, page 7-11, in the OTA chapter.

### **SSLUnknownErr**

The value for OXTAOutSSLServerPrivateKeyFile parameter in the jserv.properties file must be a PKCS-formatted key file.

If the key file is not in the correct format, the "SSLUnknownErr" error will be encountered on the client side. The server side will display the "Client certificate not provided" error.

### **SSLException X509CertExpiredErr**

**Description:** During the handshake, the program fails with SSLException and message X509CertExpiredErr. The program worked on an earlier date, and no changes have been made.

**Solution:** Your user certificate has expired. Request a new certificate from your CA.

### **SSL Exception :SSL handshake failed: X509CertChainInvalidErr**

**Description:** The handshake fails on the client side with SSLException and message X509CertChainInvalidErr. A web browser can connect to the server successfully.

**Solution:** Either the server or the client does not have the proper credentials. Ensure that in jserv.properties the OASSLCA CertFile points to a valid CA-bundle.

If the client program sets trusted certificates, you must ensure that the list includes at least one of the certificates in the server's certificate chain. In addition, you must ensure

that the server sends the complete certificate chain to the client, because Java SSL cannot build the certificate chain itself.

If you are using an Apache server, you must set the variables `SSLCertificateChainFile` and `SSLCertificateFile` correctly. This is especially important if the client program does not set trusted certificates. For more information refer to your Web server documentation.

## SSL Exception :Certificate Chain Set to SSL Context Failed

**Description:** The handshake fails on the client side with `SSLException` and message: Certificate chain set to ssl context failed.

**Solution:** If you are using intermediate CAs, you must define the CA certificates in the `ServerCACertFile` in the correct order. The file must contain the root certificate and then the intermediate certificate in that order.

## Client Connection with No Credentials

**Description:** The server lets the client connect even though no `OracleSSLCredentials` are set in the client program.

**Solution:** In order to enable security-aware applications to perform their own validation, Java SSL permits a connection if no credentials are set by the client, provided the server sends a complete certificate chain. To avoid this behavior, your application must set at least one trusted certificate.

## Oracle Diagnostic Tests

Oracle Diagnostics provides a mechanism that makes Oracle Applications more supportable and robust by providing predefined tests to check for key setup requirements. Oracle XML Gateway provides several tests through Oracle Diagnostics that you can use to check the setup of XML Gateway and review debugging information.

You can access Oracle Diagnostics through different user interfaces, including Oracle Applications Manager and other administrative consoles. For more information, see the *Oracle Applications Supportability Guide*.

The XML Gateway tests are available in Oracle Diagnostics under the Application Object Library application.

## XML Gateway Tests

The following tests are available in the XML Gateway Tests Group.

### OTA URL Validation Test

This test verifies the validity of the URLs defined for Trading Partners that are enabled for HTTP or HTTPS transport.

The test validates each URL from the output list of the OTA URL List Extraction Test, page G-37 by connecting to it using the proxy server setting. To avoid long delay, time limit of 30 seconds is added to the validation process so that the browser does not time out. If any connections fail, the test reports the problem encountered and the following trading partner information: `TP_HEADER_ID`, `TP_DETAIL_ID`, `MAP_ID`, `PROTOCOL_TYPE`, `PROTOCOL_ADDRESS`.



## OTA Round Trip Test

This diagnostic verifies that Oracle Transport Agent can successfully deliver an outbound message and receive an inbound message. The test sends a sample payload using the TestingDirectOut and TestingDirectIn message maps. The following input parameters are required:

- OTAURL
- Username/Password
- ProxyHost
- ProxyPort

If the test fails, the full response from OTA is displayed. For descriptions of the response codes, see HTTP Response Codes, page 7-22.

## Basic OTA Diagnostic Test

This test verifies the successful transmission by OTA of a sample message via SMTP, HTTP, and HTTP-ATCH. It requires the following parameters:

- Target - the URL to which OTA will send the sample message
- Email - the administrator's e-mail address
- Username/Password
- SMTP, HTTP, HTTP-ATCH - set to "YES" to test
- Time to Live - the time in seconds that the test will continue if a response has not been received

**Note:** A reasonable setting for the Time to Live parameter depends on the load in your environment.

If the test returns a successful response, but the report shows no entries for the message IDs, the Time to Live parameter was not set high enough for the test to complete. In this case you can either increase the Time to Live and resubmit the test, or you can manually check the message IDs in the Message Monitor (described below). If you choose to check manually, allow a reasonable lapse of time for the messages to be picked up by OTA.

## SSL Test

This test sends a sample message via HTTPS and verifies the certificate files and SSL parameters. Input parameters are:

- Username/Password
- Party ID/Party Site ID
- Target - the URL to which OTA will send the sample message
- Time to Live - the time in seconds that the test will continue if a response has not been received. See the note above regarding the Time to Live parameter.

## Ping Destination Server

This test verifies that the target destination is available. The input parameters are:

- Target - the URL of the server for OTA to test
- Username/Password
- ProxyHost
- ProxyPort

### **Message Monitor**

Use this test to query the status of inbound or outbound messages in the OXTA Log Message Table. This test enables you to check messages that are not queryable through the Transaction Monitor. You can query using any combination of the following criteria:

- Date
- MessageID
- Direction

You can use this test in conjunction with other diagnostic tests that create sample messages. Use the message ID from the sample message to monitor its success.

### **DTD Validation Test**

This test determines if all DTDs loaded into the database are valid. This test does not require input parameters. If invalid DTDs are found, the test returns a Failure result and lists the DTD\_ID RootElement of the invalid DTDs. To resolve, reload the invalid DTDs.

### **Map Engine Compatibility Test**

This diagnostic verifies that the map version is compatible with the XML Gateway engine version. A map is compatible with the engine if the map major version numbers are the same and the map minor version is less than or equal to the engine minor version. This test does not require input parameters.

If a map fails the test, the Map ID, Map Code, and version numbers are reported.

To resolve this error, upgrade the engine version.

### **Procedure Validity Test**

This diagnostic verifies that all user-defined procedures used by supported maps exist and are valid. This test does not require input parameters.

If the test finds an invalid procedure, the following details are reported: ProcedureFullName, Map\_ID, Map\_Code, Problem Cause, and Problem.

To resolve, review the procedure to ensure that it exists and is valid. Also verify that the defined procedure reflects the latest version of the API.

### **XSLT Validation Test**

This diagnostic verifies that all style sheets referenced by supported maps exist and are valid. The test queries the database first and if necessary queries the file system directory identified by the ECX: XSLT File Path system profile. This test does not require input parameters.

If the test finds an invalid XSLT, it returns the following details: MAP\_ID, MAP\_CODE, XSLTFileName, ProblemCause.

To resolve, reload the invalid XSLT.

## **XML Gateway Engine Test**

This diagnostic verifies the XML Gateway engine is working properly. This test does not require input parameters.

The TestingDirectOut and TestingDirectIn message maps are used in this test to analyze the following parameters:

- outbound\_ret\_code
- outbound\_errbuf
- outbound\_logfile
- inbound\_ret\_code
- inbound\_errbuf
- inbound\_logfile

If the test finds an error, the following details are returned: OutboundReturnCode, OutboundErrorMessage, OutboundLogFile, InboundReturnCode, InboundErrorMessage, and InboundLogFile.

To resolve this error, verify that all packages related to XML Gateway are valid.

## **OTA URL List Extraction Test**

This test outputs a list of URLs which are defined in the Protocol Address field in the Trading Partner Setup form for Trading Partners enabled for HTTP or HTTPS protocol type.

The output URL can be used as input to the OTA URL Validation Test for the URL validation.



---

# Index

## A

---

### Actions, 2-51

- append where clause, 2-60
- assign next sequence value, 2-59
- assign variable value, 2-58
- convert from OAG amount, 2-74
- convert from OAG datetime, 2-71
- convert from OAG OPERAMT, 2-72
- convert from OAG quantity, 2-73
- convert to OAG amount, 2-70
- convert to OAG datetime, 2-67
- convert to OAG OPERAMT, 2-67
- convert to OAG quantity, 2-69
- create global variable, 2-58
- derive target address ID from location code, 2-63
- derive target parent ID from location code, 2-64
- execute function call, 2-65
- execute procedure, 2-78
- exit program, 2-75
- get predefined variable value, 2-76
- insert into database table, 2-62
- math functions, 2-65
- perform concatenation, 2-82
- perform substring, 2-83
- send error message, 2-81
- XSLT transformation, 2-84

### Agent Listeners, 6-52

- enabling and disabling, 6-52

### API Return Codes, G-26

### API-level error messages, G-26

### APIs

- ECX\_ACTIONS.set\_error\_exit\_program, F-19
- ECX\_ATTACHMENT.formulate\_content\_id, F-24
- ECX\_ATTACHMENT.register\_attachment, F-20, F-22, F-23
- ECX\_CONDITIONS.getLengthForString, F-11
- ECX\_CONDITIONS.getPositionInString, F-11
- ECX\_CONDITIONS.getSubString, F-12
- ECX\_DOCUMENT.get\_delivery\_attrbs, F-8
- ECX\_ENG\_UTILS.convert\_from\_cxml\_datetime, F-26
- ECX\_ENG\_UTILS.convert\_to\_cxml\_date, F-25
- ECX\_ENG\_UTILS.convert\_to\_cxml\_datetime, F-25

ECX\_ERRORLOG.external\_system, F-2

ECX\_ERRORLOG.getDoclogDetails, F-17

ECX\_STANDARD.generate, 6-10

ECX\_STANDARD.getEventDetails, F-6

ECX\_STANDARD.getEventSystem, F-7

ECX\_STANDARD.getReferenceID, F-8

ECX\_STANDARD.perform\_xslt\_transformation, F-1

ECX\_STANDARD.setEventDetails, F-5

ECX\_TRADING\_PARTNER\_PVT.get\_receivers\_tp\_info, F-12

ECX\_TRADING\_PARTNER\_PVT.get\_senders\_tp\_info, F-13

ECX\_TRADING\_PARTNER\_PVT.get\_sysadmin\_email, F-14

ECX\_TRADING\_PARTNER\_PVT.getEnvelopeInformation, F-14

ECX\_TRADING\_PARTNER\_PVT.getOAGLOGICALID, F-16

how to map to, 2-86

### Application to Application integration

inbound option, 6-48

outbound option, 6-49

### Application-to-Application

example, 6-7

### Architecture, XML Gateway

diagram, 1-3

### Attachments, 2-90

OTA, 7-9

Web services, 8-2

### ATTRIBUTE3

in dynamic routing, 3-23

### ATTRIBUTE3 Field, 3-20, 4-8

### Attributes

Item type, 6-25

Workflow Item Type, 6-6

### Authentication, 7-9

client authentication, 7-9

sequence of events, 7-10

server authentication, 7-9

### AutoConfig

client authentication parameters, 7-11

## B

---

### Business Event System

- common questions, 6-54
- Business events
  - identifying seeded, 6-50
  - registering new, 6-49
- Business-to-Business
  - example, 6-8
- Buttons
  - Message Designer, 2-3
  - Message Designer Wizards, 2-15

## C

---

- Callback, 6-55, 6-55
  - how other messaging systems use, 6-57
- CBOD
  - purpose, 4-13
- Client Authentication, 7-9, 7-10
  - troubleshooting, G-33
- Client authentication, 7-9
  - AutoConfig parameters, 7-11
  - enabling, 7-11
  - parameters not set through AutoConfig, 7-12
  - setup parameters, 7-11
- Code categories, 3-29
  - seeded, B-1
- Code Conversion, 3-26
  - inbound transaction table access, 3-31
  - one to multiple external codes, 3-32
  - outbound transaction table access, 3-30
  - Standard Code Conversion Form, 3-32
  - Trading Partner Code Conversion Form, 3-34
- Code Conversion Errors, G-19
- Code Conversion Values
  - key access, 3-29
  - order of table search, 3-29
- Conditional Node Mapping, 2-38
- Confirmation BoD errors, G-21
- Confirmation Business Object Document, 4-13
  - disabling, 4-14
  - implementing, 4-14
  - seeded events and subscriptions, 4-13
  - seeded message maps, 4-13
  - structure, 4-13
- Consume XML Document Function, 6-8
- Correlation ID and Protocol Type, 3-17
- Cross-Reference Transaction Identifiers, 3-7
- Custom Messages
  - considerations for, A-17
- Custom messages
  - B2B integration, 6-53

## D

---

- Default Error Process, 6-33
- Define Lookup Values, 3-12
- Define Transactions Form, 3-7
- Define Transactions form fields, 3-8
- Define UTL\_FILE\_DIR Parameter, 3-4

- Define XML Gateway Responsibility, 3-4
- Define XML Standards Form, 3-6
- Delivery status, 6-55
- Development guidelines
  - inbound messages, 6-54
  - outbound messages, 6-53
- Diagnostics, G-34
  - XML Gateway test group, G-34
- Discontinuous Nodes
  - source definition, 2-33
  - target definition, 2-39
- Document Confirmation, 3-19
- Document Levels
  - collapsing levels, A-14
  - expanding levels, A-15
  - identifying source and target, A-13
- Downloading maps, 2-88
- DTDs
  - how to extend, 2-84
  - loading, 2-88
- Dynamic Routing, 3-23

## E

---

- E-Business Suite Application Module-Specific Item Type, 6-5
- ECX Engine Notification Process, 6-34
- ECX Event Message (FYI), 6-40
- ECX Event Notification (FYI), 6-35
- ECX External Event Message (FYI), 6-41
- ECX External Event Notification, 6-35
- ECX Main Error Process, 6-27
- ECX Main Inbound Error Process, 6-27
- ECX Main Outbound Error Process, 6-30
- ECX Reprocess Inbound Function, 6-37
- ECX Resend Outbound Message Function, 6-37
- ECX Standard Processes, 6-6
- ECX: Log File Path profile option, 3-3
- ECX: Maximum XML Size profile option, 3-3
- ECX: OAG\_LOGICAL\_ID profile option, 3-3
- ECX: Server Time Zone profile option, 3-3
- ECX: System Administrator Email Address profile option, 3-3
- ECX: XML Validate Flag profile option, 3-3
- ECX: XSLT File Path profile option, 3-3
- ECX\_ACTIONS, F-19
- ECX\_ATTACHMENT, F-20
- ECX\_CONDITIONS, F-11
- ECX\_DOCLOGS table, 6-25, 6-28, 6-37, 6-37
- ECX\_DOCUMENT, F-8
- ECX\_ENG\_UTILS, F-25
- ECX\_ERRORLOG, F-2, F-17
- ECX\_IN\_OAG\_Q, 5-1
- ECX\_INBOUND queue, 5-1
- ECX\_OUTBOUND Queue, 5-1
- ECX\_STANDARD, F-5
- ECX\_TRADING\_PARTNER\_PVT, F-12
- ECXOTAINbound.html File, 5-3

- ecxver.sql script, G-1
- EDI Transactions
  - compared to XML messages, 1-8
- Element Mapping, 2-48
  - applying actions, 2-50
  - guidelines, 2-49
  - icons, 2-50
- Enable User Security profile option, 3-3
- Engine-Level Error Messages
  - error code, G-12
  - error type, G-11
- Enqueue a message with Non-Oracle messaging system, 5-5
- Enqueue/Dequeue Errors, G-19
- Envelope, 4-6
  - components of, 4-6
- Error Handling
  - for inbound messages, 6-28
  - for outbound messages, 6-30
- Error Handling Processes, 6-25
  - relationship, 6-26
- Error Messages
  - engine-level, G-11
- Error notifications, 6-50
- Error Queue, 5-3
- Event subscriptions
  - configuring, 6-50
  - deleting, 6-50
- Events
  - XML Gateway Error Processing Item Type, 6-38
    - Message Delivery Error, 6-39
    - Receive Error, 6-39
    - Receive Send Notification, 6-39
    - Receive the inbound subscription error, 6-39
    - Receive the inbound subscription processing error, 6-39
  - XML Gateway Standard Item Type, 6-19
    - Generic Receive Event, 6-20
    - Raise Document Delivery Event, 6-20
    - WF Send, 6-22
- Execution Engine
  - functions, 1-7
  - process flow, 4-1
  - required setup, 4-2
  - trading partner validation for inbound messages, 4-9
  - trading partner validation for outbound messages, 4-12
- Execution engine
  - APIs, F-1
- Execution Engine Processing Sequence, 2-52

## F

---

- Forms
  - Define Lookup Values, 3-12
  - Define Transactions, 3-7

- Define XML Standards, 3-6
- Hub Definitions, 3-5
- Standard Code Conversion, 3-32
- Trading Partner Code Conversion, 3-34
- Trading Partner Setup, 3-13
- Functions (Workflow)
  - XML Gateway Error Processing Item Type, 6-36
    - ECX Reprocess Inbound, 6-37
    - ECX Resend Outbound Message, 6-37
    - Get ECX In Error Details, 6-37
    - Get ECX Out Error Details, 6-37
    - Get System Administrator Role, 6-37
    - Get Trading Partner Role, 6-38
  - XML Gateway Standard Item Type, 6-7
    - Consume XML Document, 6-8
    - Generate Trading Partner XML Document, 6-12
    - Generate XML Document, 6-9
    - Send Document, 6-14
    - Transaction Delivery Required, 6-16
- FYI Delivery Error Process, 6-32

## G

---

- General processing errors, G-24
- Generate Trading Partner XML Document, 6-12
- Generate XML Document Function, 6-9
- Generic Receive Event, 6-20
- Get ECX In Error Details Function, 6-37
- Get ECX Out Error Details Function, 6-37
- Get System Administrator Role Function, 6-37
- Get Trading Partner Role Function, 6-38

## H

---

- HTTP status codes, 7-22
- Hub Definitions Form, 3-5
- Hub Entity Code, 3-5
- Hub Errors, G-18

## I

---

- Implementation Checklist, 3-1
- Inbound Message Queue, 5-1, 5-2
- inbound messages
  - required communications data, 3-20
- Inbound processing errors, G-24
- Inbound Trading Partner Message, 6-41
- Inbound Transaction Queue, 5-1, 5-2
- Item Type
  - components of, 6-4
  - identifying seeded, 6-50
- Item Types
  - E-Business Suite Application Module-Specific Item Type, 6-5
  - XML Gateway Error Processing Item Type, 6-5
  - XML Gateway Standard Item Type, 6-4, 6-5

## **L**

---

- Level Mapping Guidelines
  - collapsing levels, 2-45
  - discontinuous nodes, 2-47, A-16
  - expanding levels, 2-46
  - OAG DTDs, 2-45
- Level Mapping Tab, 2-44
- Lookup Types
  - XML Gateway Error Processing Item Type
    - ECX Inbound Error Actions, 6-44
    - ECX Outbound Error Actions, 6-44
    - ECX Outbound Error Types, 6-44
  - XML Gateway Standard Item Type
    - send mode, 6-24
- Lookup Values
  - defining, 3-12

## **M**

---

- Map Action Editor, 2-53
  - action level and action stage combinations, 2-54
  - invoking, 2-53
- Map analysis
  - compare database views to DTD, A-9
  - supported actions, C-1
- Map Analysis Guidelines
  - inbound messages, A-7
  - outbound messages, A-1
- Map analysis guidelines
  - comparing database views to DTD, A-3
  - inbound message checklist, A-7
- Map Analysis Guidelines for outbound messages, A-1
- Map Analysis overview, A-1
- Maps
  - downloading, 2-88
- Menus
  - Message Designer
    - help menu, 2-3
- Menus
  - Message Designer, 2-2
- Menus, Message Designer
  - file menu, 2-2
  - Message Designer
    - File Menu, properties option, 2-4
    - view menu, 2-2
- Message Delivery Callback, 6-55
  - block mode, 6-56
- Message delivery callback, 6-57
  - how other messaging systems use, 6-57
- Message Delivery Error (FYI), 6-35
- Message Delivery Error Event, 6-39
- Message Designer
  - APIs, F-3
- Message Envelope, 4-6
- Message Format, 5-4
- Message map and DTD errors, G-21

- Message Maps
  - loading, 2-87
- Messages
  - development guidelines for outbound, 6-53
  - XML Gateway Error Processing Item Type, 6-40
    - ECX Event Message (FYI), 6-40
    - ECX External Event Message (FYI), 6-41
    - Inbound Trading Partner Message, 6-41
    - message template attributes, 6-41
    - Outbound Trading Partner Message, 6-41
- messages
  - development guidelines inbound, 6-54
  - inbound
    - required communications data, 3-20
  - outbound
    - required communications data, 3-19
- Multi-Org Consideration, 3-14

## **N**

---

- Naming Conventions
  - transaction type and transaction subtype, 3-9
- Naming conventions, D-1
- Non-OTA servers
  - connecting to, 7-13
- Notifications
  - XML Gateway Error Processing Item Type, 6-35
    - ECX Event Notification (FYI), 6-35
    - ECX External Event Notification, 6-35
    - Message Delivery Error (FYI), 6-35
    - Trading Partner Inbound Error Notification, 6-36
    - Trading Partner Outbound Error Notification, 6-36
- Noun
  - setting in OAG standards, 3-11

## **O**

---

- OAG, 1-1
  - recommending changes, A-16
- OAG conversions table, 2-66
- Oracle Transport Agent (OTA), 7-1
- OTA
  - attachments, 2-91
  - code connection samples, 7-14
    - connection over HTTP, 7-14
    - connection over HTTPS, 7-18
  - troubleshooting, 7-22
- OTA message propagation flow, 7-2
- OTA Protocol, 7-2
- Outbound Message Queue, 5-1, 5-2
- outbound messages
  - required communications data, 3-19
- Outbound Trading Partner Message, 6-41
- overview, 1-1



## P

---

- Pass-Through Messages
  - execution engine processing of, 4-10
  - how to map, 2-86
- Post message
  - OTA, 7-3
- Processes
  - XML Gateway Error Processing Item Type, 6-25
  - XML Gateway Standard Item Type, 6-6
- Profile Options, 3-2
  - ECX: Log File Path, 3-3
  - ECX: Maximum XML Size, 3-3
  - ECX: OAG\_LOGICALID, 3-3
  - ECX: Server Time Zone, 3-3
  - ECX: System Administrator Email Address, 3-3
  - ECX: XML Validate Flag, 3-3
  - ECX: XSLT File Path, 3-3
  - Enable User Security, 3-3
- Properties Menu Option, 2-4
- Protocol Type
  - and correlation ID, 3-17
  - in Trading Partner setup, 3-17
- Protocol types, 4-4
- Purging transaction, 6-50

## Q

---

- Queues
  - definition, 5-1
  - error, 5-3
  - inbound, 5-2
  - inbound message, 5-2
  - inbound transaction, 5-2
  - outbound, 5-2
  - outbound message, 5-2
  - used by XML Gateway, 5-1

## R

---

- Raise Document Delivery Event, 6-20
- Receive Error Event, 6-39
- Receive Send Notification Event, 6-39
- Receive the inbound subscription error event, 6-39
- Receive the inbound subscription processing error event, 6-39
- Reprocess, 6-25
- Response message
  - OTA, 7-8
- Retry, 6-25, 6-37
- Routing, 3-19
  - dynamic, 3-23
  - static, 3-21
  - static and dynamic, 3-20
- Rule Function, Workflow Default, 6-2
- Rule Function, XML Gateway, 6-2

## S

---

- Send Document Function, 6-14
- Server Authentication, 7-9
- SOAP, 8-1
- SOAP servlet, 8-2
- Source Definition Overview, 2-33
- Source Definition Tab, 2-35
- SSL
  - how to configure for Web services, 8-5
- Standard Code Conversion Form, 3-32
- standards used by XML Gateway, 1-1
- Static and Dynamic Routing, 3-20
- Static Routing, 3-21
- Stylesheets
  - loading and deleting, 2-89
- Supported actions
  - map analysis, C-1
- System Profile Options, 3-2
- system.ecxmsg data type, 5-4

## T

---

- Target Definition, 2-39
- Target Definition Tab, 2-41
- Time zones
  - set up, 3-3
  - values for profile option, E-1
- Toolbar
  - Message Designer, 2-3
- Trace, 6-51
- Trading Partner Code Conversion Form, 3-34
- Trading Partner Errors, G-14
- Trading Partner Inbound Error Notification, 6-36
- Trading Partner Outbound Error Notification, 6-36
- Trading Partner Setup Form, 3-13
  - data validation, 4-11
- Trading Partner User Security, 3-24
- Trading Partner Validation
  - inbound messages, 4-9
  - outbound messages, 4-12
- Transaction Delivery Required Function, 6-16
- Transaction Monitor, G-5
  - calling from other applications, G-6
  - Details page, G-9
- Transaction status
  - monitoring, 6-52
- Troubleshooting
  - API-level messages, G-26
  - client authentication issues, G-33
  - engine-level error messages, G-11
  - engine-level errors
    - code conversion errors, G-19
    - Confirmation BoD errors, G-21
    - enqueue/dequeue errors, G-19
    - general processing errors, G-24
    - hub errors, G-18
    - inbound processing errors, G-24

- message map and DTD errors, G-21
- trading partner errors, G-14
- UTL\_FILE errors, G-14
- Workflow Business Event System errors, G-20
- HTTP Status Codes, 7-22
- manual troubleshooting steps, G-11
- running the ecxver.sql script, G-1
- transaction monitor, G-5
- version validation, G-31

## U

---

- Using JMS Queues
  - Custom Inbound JMS Queues, 9-12
  - Custom JMS Queues, 9-5
  - Custom Outbound JMS Queues, 9-5
  - Integration Points, 9-2
  - JMS Integration, 9-3
  - Overview, 9-1
- UTL\_FILE errors, G-14

## V

---

- Validation
  - trading partner for inbound messages, 4-9
  - trading partner for outbound messages, 4-12
- Verb
  - setting in OAG standards, 3-11
- Version validation, G-31

## W

---

- Web services, 8-1
  - attachments, 8-2
  - components, 8-1
  - diagnostic tests
    - class loader test, 8-6
    - parameter test, 8-6
    - round trip test, 8-6
  - diagnostics, 8-6
  - example, 8-7
    - prerequisite files, 8-7
  - example scripts, 8-8
  - Prerequisite files, 8-9
  - process flow, 8-3
    - inbound, 8-3
    - outbound, 8-3
  - setup steps, 8-4
  - SOAP servlet, 8-2
  - WSDL, 8-1
- WF Send Event, 6-22
- WF\_ERROR Queue, 5-1
- Wizards
  - data definition creation, 2-12
  - map creation, 2-13
- Workflow Builder
  - components of an item type, 6-4

- E-Business Suite Application Module-Specific Item Type, 6-5
- item types, 6-3
- XML Gateway Error Processing Item Type, 6-5
- XML Gateway Standard Item Type, 6-4, 6-5
- Workflow Business Event System
  - overview of integration, 6-1
- Workflow Business Event System errors, G-20
- Workflow Default Rule Function
  - and Trading Partner validation for outbound messages, 4-12
- Workflow Error Queue, 5-1, 5-3
- Workflow Processes
  - monitoring, 6-51
- WSDL, 8-1
  - schema structure type, 8-1

## X

---

- XML Gateway Error Processing Item Type, 6-5, 6-24
  - events, 6-38
    - Message Delivery Error, 6-39
    - Receive Error, 6-39
    - Receive Send Notification, 6-39
    - Receive the inbound subscription error, 6-39
    - Receive the inbound subscription processing error, 6-39
  - functions, 6-36
    - ECX Reprocess Inbound, 6-37
    - ECX Resend Outbound Message, 6-37
    - Get ECX In Error Details, 6-37
    - Get ECX Out Error Details, 6-37
    - Get System Administrator Role, 6-37
    - Get Trading Partner Role, 6-38
  - lookup types, 6-43
    - ECX Inbound Error Actions, 6-44
    - ECX Outbound Error Actions, 6-44
    - ECX Outbound Error Types, 6-44
  - message template attributes, 6-41
  - messages, 6-40
    - ECX Event Message (FYI), 6-40
    - ECX External Event Message (FYI), 6-41
    - Inbound Trading Partner Message, 6-41
    - Outbound Trading Partner Message, 6-41
  - notifications, 6-35
    - ECX Event Notification (FYI), 6-35
    - ECX External Event Notification, 6-35
    - Message Delivery Error (FYI), 6-35
    - Trading Partner Inbound Error Notification, 6-36
    - Trading Partner Outbound Error Notification, 6-36
  - processes
    - Default Error Process, 6-33
    - ECX Engine Notification Process, 6-34
    - ECX Main Error Process, 6-27
    - ECX Main Inbound Error Process, 6-27

- ECX Main Outbound Error Process, 6-30
- Error Handling for Inbound Messages, 6-28
- Error Handling for Outbound Messages, 6-30
- FYI Delivery Error Process, 6-32
- XML Gateway Rule Function
  - and Trading Partner validation for outbound messages, 4-12
- XML Gateway Standard Item Type, 6-4, 6-5
  - events, 6-19
- Generic Receive Event, 6-20
- Raise Document Delivery Event, 6-20
- WF Send, 6-22
  - functions, 6-7
  - lookup types, 6-22
  - send mode, 6-24
- XML message
  - reviewing outbound, 6-52
- XSLT stylesheets
  - loading and deleting, 2-89

