Oracle® Workflow

Developer's Guide

Release 2.6.4

Part No. B15853-01

June 2005

ORACLE

Oracle Workflow Developer's Guide, Release 2.6.4

Part No. B15853-01

Copyright © 2003, 2005, Oracle. All rights reserved.

Primary Author: Siu Chang, Clara Jaeckel

Contributing Author: Varsha Bhatia, George Buzsaki, John Cordes, Mark Craig, Avinash Dabholkar, Mark Fisher, Yongran Huang, Kevin Hudson, George Kellner, Sai Kilaru, Angela Kung, David Lam, Janet Lee, Jin Liu, Kenneth Ma, Steve Mayze, Santhana Natarajan, Rajesh Raheja, Varadarajan Rajaram, Tim Roveda, Robin Seiden, Vijay Shanmugam, Sachin Sharma, Sheryl Sheh, Allison Sparshott, Susan Stratton, Roshin Thomas, Robert Wunderlich

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 Overview of Oracle Workflow

Overview of Oracle Workflow for Developers 1-1
Major Features and Definitions 1-2
Workflow Processes 1-4

2 Defining a Workflow Process

Overview of Oracle Workflow Builder 2-1
The Navigator Tree Structure 2-2
Viewing the Navigator Tree 2-3
Creating Process Definitions in Oracle Workflow Builder 2-5
Opening and Saving Item Types 2-9
Quick Start Wizard Overview 2-14
Using Oracle Workflow Builder with Different Server Versions 2-15
Item Type Definition Web Page 2-18

3 Defining Workflow Process Components

Workflow Process Components 3-1
Item Types 3-2
Attributes 3-5
Defining Item Types and Attributes 3-6
Allowing Access to an Object 3-17
Lookup Types 3-19
Defining Lookup Types 3-20
Messages 3-22
Defining Messages 3-42
Activities 3-52
Defining Activities 3-58
Voting Activity 3-69
Example Voting Methods 3-72
Deleting Objects in Oracle Workflow Builder 3-74

Modifying Objects in Oracle Workflow Builder . . . 3-75
Workflow Objects That Support Versioning . . . 3-76
Workflow Objects That Do Not Support Versioning . . . 3-76

4 Defining a Workflow Process Diagram

Process Window . . . 4-1
Diagramming a Process . . . 4-3
Modifying Fonts in Oracle Workflow Builder . . . 4-18
Creating a Shortcut Icon for a Workflow Process . . . 4-18
Referencing Roles . . . 4-19
Initiating a Process . . . 4-22

5 Predefined Workflow Activities

Standard Activities . . . 5-1
And/Or Activities . . . 5-2
Comparison Activities . . . 5-2
Compare Execution Time Activity . . . 5-3
Wait Activity . . . 5-3
Block Activity . . . 5-4
Defer Thread Activity . . . 5-4
Launch Process Activity . . . 5-5
Noop Activity . . . 5-5
Loop Counter Activity . . . 5-5
Start Activity . . . 5-7
End Activity . . . 5-7
Role Resolution Activity . . . 5-7
Notify Activity . . . 5-8
Vote Yes/No Activity . . . 5-8
Master/Detail Coordination Activities . . . 5-9
Assign Activity . . . 5-12
Get Monitor URL Activity . . . 5-12
Get Event Property Activity . . . 5-12
Set Event Property Activity . . . 5-13
Compare Event Property Activity . . . 5-13
XML Get Tag Value Activity . . . 5-14
XML Compare Tag Value Activities . . . 5-15
XML Transform Activity . . . 5-16
Concurrent Manager Standard Activities . . . 5-17
Execute Concurrent Program Activity . . . 5-17
Submit Concurrent Program Activity . . . 5-18
Wait for Concurrent Program Activity . . . 5-19

6 Defining Procedures and Functions for Oracle Workflow

Defining Procedures and Functions for Oracle Workflow . . . 6-1

Standard API for PL/SQL Procedures Called by Function Activities . . . 6-2
Standard API for Java Procedures Called by Function Activities . . . 6-7
Standard API for an Item Type Selector or Callback Function . . . 6-10
Standard APIs for "PL/SQL" Documents . . . 6-13
"PL/SQL" Documents . . . 6-13
"PL/SQL CLOB" Documents . . . 6-15
"PL/SQL BLOB" Documents . . . 6-18
Standard API for an Event Data Generate Function . . . 6-19
Standard API for a PL/SQL Generate Function . . . 6-20
Standard API for a Java Generate Function . . . 6-20
Standard APIs for a Queue Handler . . . 6-21
Standard APIs for a PL/SQL Queue Handler . . . 6-21
Standard APIs for a Java Queue Handler . . . 6-22
Standard API for an Event Subscription Rule Function . . . 6-23
Standard API for a PL/SQL Subscription Rule Function . . . 6-25
Standard API for a Java Subscription Rule Function . . . 6-26

7 Testing a Workflow Definition

Testing Workflow Definitions . . . 7-1
Testing Workflow Definitions Using the Developer Studio . . . 7-2

8 Managing Business Events

Managing Business Events . . . 8-2
Event Manager for Standalone Oracle Workflow . . . 8-2
Events . . . 8-3
Defining Events . . . 8-5
Systems . . . 8-12
Defining Systems . . . 8-13
Agents . . . 8-17
Defining Agents . . . 8-25
Event Subscriptions . . . 8-29
Defining Event Subscriptions . . . 8-38
Raising Events . . . 8-44
Signing Up Systems . . . 8-45
Synchronizing Systems . . . 8-47
Event Manager for Oracle Applications . . . 8-49
Events . . . 8-50
Defining Events . . . 8-53
Event Subscriptions . . . 8-58
Defining Event Subscriptions . . . 8-70
Agents . . . 8-76
Defining Agents . . . 8-88
Systems . . . 8-92
Defining Systems . . . 8-96
Workflow Agent Ping/Acknowledge . . . 8-99

The Workflow Agent Ping/Acknowledge Item Type . . . 8-100
Summary of the Master Ping Process . . . 8-101
Master Ping Process Activities . . . 8-102
Summary of the Detail Ping Process . . . 8-103
Detail Ping Process Activities . . . 8-103

9 Predefined Workflow Events

Predefined Workflow Events . . . 9-1
Event Definition Events . . . 9-2
Event Group Definition Events . . . 9-3
System Definition Events . . . 9-4
Agent Definition Events . . . 9-5
Agent Group Definition Events . . . 9-7
Event Subscription Definition Events . . . 9-8
Synchronize Event Systems Event . . . 9-9
Seed Event Group . . . 9-9
Ping Agent Events . . . 9-12
System Signup Event . . . 9-13
Any Event . . . 9-14
Unexpected Event . . . 9-17
User Entry Has Changed Event . . . 9-19
Notification Events . . . 9-21
Notification Mailer Events . . . 9-30
Business Event System Control Events . . . 9-32
Generic Service Component Framework Control Events . . . 9-34
Workflow Engine Events . . . 9-37
Directory Service Events . . . 9-38
Workflow Send Protocol . . . 9-45
The Workflow Send Protocol Item Type . . . 9-46
Summary of the Workflow Event Protocol Process . . . 9-47
Workflow Event Protocol Process Activities . . . 9-49
Workflow Send Protocol Events . . . 9-51

10 Demonstration Workflow Processes

Sample Workflow Processes . . . 10-1
Displaying the Process Diagram of a Sample Workflow . . . 10-2
Requisition Process . . . 10-3
Installing the Requisition Data Model . . . 10-4
Initiating the Requisition Workflow . . . 10-6
The Requisition Item Type . . . 10-8
Summary of the Requisition Approval Process . . . 10-9
Requisition Process Activities . . . 10-11
Summary of the Notify Approver Subprocess . . . 10-15
Notify Approver Subprocess Activities . . . 10-16
Sample StartProcess Function . . . 10-18

Example Function Activities . . . 10-21
Example: Select Approver . . . 10-21
Example: Verify Authority . . . 10-24
Example Notification Activity . . . 10-26
Example: Notify Requisition Approval Required . . . 10-26
Event System Demonstration . . . 10-28
Installing the Event System Demonstration Data Model . . . 10-30
Initiating the Event System Demonstration Workflow . . . 10-31
The Event System Demonstration Item Type . . . 10-34
Summary of the Buyer: Top Level PO Process . . . 10-36
Buyer: Top Level PO Process Activities . . . 10-37
Summary of the Buyer: Send PO to Supplier Subprocess . . . 10-40
Buyer: Send PO to Supplier Subprocess Activities . . . 10-40
Summary of the Buyer: Receive Supplier PO Acknowledgement Subprocess . . . 10-42
Buyer: Receive Supplier PO Acknowledgement Subprocess Activities . . . 10-42
Summary of the Buyer: Advanced Shipment Notice Subprocess . . . 10-44
Buyer: Advanced Shipment Notice Subprocess Activities . . . 10-44
Summary of the Buyer: Receive Supplier Invoicing Subprocess . . . 10-46
Buyer: Receive Supplier Invoicing Subprocess Activities . . . 10-46
Summary of the Supplier: Top Level Order Process . . . 10-47
Supplier: Top Level Order Process Activities . . . 10-48
Summary of the Supplier: Get Order Details Subprocess . . . 10-51
Supplier: Get Order Details Subprocess Activities . . . 10-52
Summary of the Supplier: Credit Check Subprocess . . . 10-54
Supplier: Credit Check Subprocess Activities . . . 10-55
Summary of the Supplier: Stock Check Subprocess . . . 10-56
Supplier: Stock Check Subprocess Activities . . . 10-57
Summary of the Supplier: Advanced Shipment Notice Subprocess . . . 10-58
Supplier: Advanced Shipment Notice Subprocess Activities . . . 10-58
Summary of the Supplier: Send Supplier Invoice Subprocess . . . 10-59
Supplier: Send Supplier Invoice Subprocess Activities . . . 10-60
B2B Purchase Order Event . . . 10-61
B2B Purchase Order Acknowledgement Event . . . 10-64
B2B Advanced Shipment Notice Event . . . 10-65
B2B Invoice Event . . . 10-67

11 Error Handling

Error Handling . . . 11-1
Error Handling for Workflow Processes . . . 11-1
Error Handling for Event Subscription Processing . . . 11-3
System: Error Item Type and Item Attributes . . . 11-5
Default Error Process . . . 11-6
Retry-only Process . . . 11-9
Default Event Error Process . . . 11-11

A Oracle Workflow Developer Navigation Paths

Oracle Workflow Developer Navigation Paths A-1

B Oracle Workflow Builder Menus and Toolbars

Oracle Workflow Builder Menus B-1
Oracle Workflow Builder Toolbars B-4

C Oracle Workflow Implementation in Other Oracle Products

Predefined Workflows Embedded in Oracle E-Business Suite C-1
Advanced Planning. C-1
Applied Technology C-3
Business Intelligence C-6
Communications C-6
Contracts and Service C-6
Financial Applications C-7
Higher Education C-15
HRMS Applications C-20
Leasing C-25
Maintenance Applications C-29
Manufacturing Applications C-29
Order Fulfillment C-34
Procurement. C-36
Projects C-37
Sales, Marketing, and eCommerce C-38
Oracle Workflow Business Event System Implementation in Oracle E-Business Suite . . . C-40
Oracle Workflow Implementation in the Oracle Technology Stack C-41
Oracle Workflow Support Policy C-42
Customization Guidelines C-42
Resolving Customization Issues C-43
What Is NOT Supported C-43
What Is Supported C-44

Glossary

Index

Send Us Your Comments

Oracle Workflow Developer's Guide, Release 2.6.4

Part No. B15853-01

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

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

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

- Electronic mail: 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 2.6.4 of the *Oracle Workflow Developer's Guide*.

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

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

  If you have never used Oracle Workflow, 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*.
- Operating system concepts.
- Oracle Database, Oracle Application Server, and PL/SQL technology.

  If you have never used these products, Oracle suggests you attend training classes available through Oracle University.

See Related Documents on page xiii 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 Overview of Oracle Workflow

This chapter introduces you to the concept of a workflow process and to the major features of Oracle Workflow.

2 Defining a Workflow Process

This chapter tells you how to use Oracle Workflow Builder to define a workflow process definition.

3 Defining Workflow Process Components

This chapter tells you how to use Oracle Workflow Builder to define the components necessary to compose a workflow process diagram.

4 Defining a Workflow Process Diagram

This chapter tells you how to use Oracle Workflow Builder to define a workflow process diagram and how to load roles from the database so you can assign notification activities to specific roles.

5 Predefined Workflow Activities

This chapter tells you how to use Oracle Workflow's predefined activities.

6 Defining Procedures and Functions for Oracle Workflow

This chapter describes the standard APIs to use for Oracle Workflow PL/SQL and Java procedures and functions.

7 Testing a Workflow Definition

This chapter tells you how to test your workflow definitions using the standalone Launch Processes Web page or Oracle Applications Developer Studio.

8 Managing Business Events

This chapter tells you how to manage business events using the Oracle Workflow Event Manager Web pages.

9 Predefined Workflow Events

This chapter tells you how to use Oracle Workflow's predefined events.

10 Demonstration Workflow Processes

This chapter describes the demonstration workflow processes provided with Oracle Workflow. These demonstration processes showcase many Oracle Workflow features.

11 Error Handling

This chapter describes how Oracle Workflow handles errors in workflow processes and event subscription processing.

A Oracle Workflow Developer Navigation Paths

This appendix lists the navigation paths to Oracle Workflow developer Web pages in the seeded Oracle Workflow responsibilities for Oracle Applications.

B Oracle Workflow Builder Menus and Toolbars

This appendix provides a description of the menus and toolbars in Oracle Workflow Builder.

C Oracle Workflow Implementation in Other Oracle Products

This appendix lists embedded workflows and Business Event System implementation in Oracle E-Business Suite and the Oracle technology stack, as well as Oracle's support policy towards the customization of these workflows, events, and subscriptions.

Glossary

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 Workflow.

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 Workflow 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 Oracle*MetaLink* and is updated frequently.
- **Online Help** - Online help patches (HTML) are available on Oracle*MetaLink*.
- **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 Oracle*MetaLink*.

If you are using the standalone version of Oracle Workflow, 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 Workflow Web pages.

Related Guides

You may want to refer to other Oracle Workflow guides and Oracle Applications implementation documentation when you set up and use Oracle Workflow. Additionally, Oracle Workflow is used by other Oracle Applications products to provide embedded workflows and business events. Therefore, if you are using the version of Oracle Workflow embedded in Oracle Applications, you may want to refer to other products' guides to learn more about the workflows and business events they include.

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.

Oracle Workflow Documentation Set

Oracle Workflow Administrator's Guide

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

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 Workflow API Reference

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

Guides Related to This Product

Oracle Assets User Guide

In Oracle Assets, you can post capital project costs to become depreciable fixed assets. Refer to this guide to learn how to query mass additions imported from other products to Oracle Assets and to review asset information.

Oracle General Ledger User Guide

Use this manual when you plan and define your chart of accounts, accounting period types and accounting calendar, functional currency, and set of books. The manual also describes how to define journal entry sources and categories so you can create journal entries for your general ledger. If you use multiple currencies, use this manual when you define additional rate types, and enter daily rates. This manual also includes complete information on implementing Budgetary Control.

Oracle HRMS Documentation Set

This set of guides explains how to define your employees, so you can give them operating unit and job assignments. It also explains how to set up an organization (operating unit). Even if you do not install Oracle HRMS, you can set up employees and

organizations using Oracle HRMS windows. Specifically, the following manuals will help you set up employees and operating units:

- ***Using Oracle HRMS - The Fundamentals***

  This user guide explains how to set up and use enterprise modeling, organization management, and cost analysis.

- ***Managing People Using Oracle HRMS***

  Use this guide to learn about entering employees.

Oracle Payables User Guide

Refer to this manual to learn how to use Invoice Import to create invoices in Oracle Payables from expense reports data in the Oracle Payables interface tables. This manual also explains how to define suppliers, and how to specify supplier and employee numbering schemes for invoices.

Oracle Projects Implementation Guide

Use this manual as a guide for implementing Oracle Projects. This manual also includes appendixes covering function security, menus and responsibilities, and profile options.

Oracle Purchasing User Guide

Use this guide to learn about entering and managing the requisitions and purchase orders that relate to your projects. This manual also explains how to create purchase orders from project-related requisitions in the AutoCreate Documents window.

Oracle Receivables User Guide

Use this manual to learn more about Oracle Receivables invoice processing and invoice formatting, defining customers, importing transactions using AutoInvoice, and defining automatic accounting in Oracle Receivables.

Oracle Business Intelligence System Implementation Guide

This guide provides information about implementing Oracle Business Intelligence (BIS) in your environment

BIS 11i User Guide Online Help

This guide is provided as online help only from the BIS application and includes information about intelligence reports, Discoverer workbooks, and the Performance Management Framework.

Using Oracle Time Management

This guide provides information about capturing work patterns such as shift hours so that this information can be used by other applications such as Oracle General Ledger.

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 installing Oracle 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 and the Oracle 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 Alert User's Guide

This guide explains how to define periodic and event alerts to monitor the status of your Oracle Applications data.

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 11*i*. 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 Applications. This manual details additional steps and setup considerations for implementing Oracle Applications with this feature.

Multiple Organizations in Oracle Applications

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

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 guide also provides information on creating custom reports on flexfields data.

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 forms so that they integrate with Oracle Applications.

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 forms-based Oracle Applications products and how to apply this UI to the design of an application built using Oracle Forms.

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 Oracle*MetaLink*.

Oracle Applications Message Reference Manual

This manual describes 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 Workflow 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 Workflow 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.

1

Overview of Oracle Workflow

This chapter introduces you to the concept of a workflow process and to the major features of Oracle Workflow.

This chapter covers the following topics:

- Overview of Oracle Workflow for Developers

Overview of Oracle Workflow for Developers

Oracle Workflow delivers a complete workflow management system that supports business process based integration. Its technology enables modeling, automation, and continuous improvement of business processes, routing information of any type according to user-defined business rules.

E-business is accelerating the demand for integration of applications within the enterprise as well as integration of a company's systems with trading partners and business-to-business exchanges. Oracle Workflow automates and streamlines business processes both within and beyond your enterprise, supporting traditional applications based workflow as well as e-business integration workflow. Oracle Workflow is unique in providing a workflow solution for both internal processes and business process coordination between applications.

Routing Information

Business processes today involve getting many types of information to multiple people according to rules that are constantly changing. With so much information available, and in so many different forms, how do you get the right information to the right people? Oracle Workflow lets you provide each person with all the information they need to take action. Oracle Workflow can route supporting information to each decision maker in a business process, including people both inside and outside your enterprise.

Defining and Modifying Business Rules

Oracle Workflow lets you define and continuously improve your business processes using a drag-and-drop process designer.

Unlike workflow systems that simply route documents from one user to another with some approval steps, Oracle Workflow lets you model sophisticated business processes. You can define processes that loop, branch into parallel flows and then rendezvous, decompose into subflows, and more. Because Oracle Workflow can decide which path to take based on the result of a stored procedure, you can use the power of

Java and of PL/SQL, the language of the Oracle Database, to express any business rule that affects a workflow process. See: Workflow Processes, page 1-4.

Delivering Electronic Notifications

Oracle Workflow extends the reach of business process automation throughout the enterprise and beyond to include any e-mail or Internet user. Oracle Workflow lets people receive notifications of items awaiting their attention via e-mail, and act based on their e-mail responses. You can even view your list of things to do, including necessary supporting information, and take action using a standard Web browser.

Integrating Systems

Oracle Workflow lets you set up subscriptions to business events which can launch workflows or enable messages to be propagated from one system to another when business events occur. You can communicate events among systems within your own enterprise and with external systems as well. In this way, you can implement point-to-point messaging integration or use Oracle Workflow as a messaging hub for more complex system integration scenarios. You can model business processes that include complex routing and processing rules to handle events powerfully and flexibly.

Major Features and Definitions

Oracle Workflow Builder

Oracle Workflow Builder is a graphical tool that lets you create, view, or modify a business process with simple drag and drop operations. Using the Workflow Builder, you can create and modify all workflow objects, including activities, item types, and messages. See: Workflow Processes, page 1-4.

At any time you can add, remove, or change workflow activities, or set up new prerequisite relationships among activities. You can easily work with a summary-level model of your workflow, expanding activities within the workflow as needed to greater levels of detail. And, you can operate Oracle Workflow Builder from a desktop PC or from a disconnected laptop PC.

Workflow Engine

The Workflow Engine embedded in the Oracle Database implements process definitions at runtime. The Workflow Engine monitors workflow states and coordinates the routing of activities for a process. Changes in workflow state, such as the completion of workflow activities, are signaled to the engine via a PL/SQL API or a Java API. Based on flexibly-defined workflow rules, the engine determines which activities are eligible to run, and then runs them. The Workflow Engine supports sophisticated workflow rules, including looping, branching, parallel flows, and subflows.

Business Event System

The Business Event System is an application service that uses the Oracle Advanced Queuing (AQ) infrastructure to communicate business events between systems. The Business Event System consists of the Event Manager, which lets you register subscriptions to significant events, and event activities, which let you model business events within workflow processes.

When a local event occurs, the subscribing code is executed in the same transaction as the code that raised the event. Subscription processing can include executing custom code on the event information, sending event information to a workflow process, and sending event information to other queues or systems.

Workflow Definitions Loader

The Workflow Definitions Loader is a utility program that moves workflow definitions between database and corresponding flat file representations. You can use it to move workflow definitions from a development to a production database, or to apply upgrades to existing definitions. In addition to being a standalone server program, the Workflow Definitions Loader is also integrated into Oracle Workflow Builder, allowing you to open and save workflow definitions in both a database and file.

Complete Programmatic Extensibility

Oracle Workflow lets you include your own PL/SQL procedures or external functions as activities in your workflows. Without modifying your application code, you can have your own program run whenever the Workflow Engine detects that your program's prerequisites are satisfied.

Electronic Notifications

Oracle Workflow lets you include users in your workflows to handle activities that cannot be automated, such as approvals for requisitions or sales orders. The Notification System sends notifications to and processes responses from users in a workflow. Electronic notifications are routed to a role, which can be an individual user or a group of users. Any user associated with that role can act on the notification.

Each notification includes a message that contains all the information a user needs to make a decision. The information may be embedded in the message body or attached as a separate document. Oracle Workflow interprets each notification activity response to decide how to move on to the next workflow activity.

Electronic Mail Integration

Electronic mail (e-mail) users can receive notifications of outstanding work items and can respond to those notifications using their e-mail application of choice. An e-mail notification can include an attachment that provides another means of responding to the notification.

Internet-Enabled Workflow

Any user with access to a standard Web browser can be included in a workflow. Web users can access a Notification Web page to see their outstanding work items, then navigate to additional pages to see more details or provide a response.

Monitoring and Administration

Workflow administrators and users can view the progress of a work item in a workflow process by connecting to the Workflow Monitor using a standard Web browser that supports Java. The Workflow Monitor displays an annotated view of the process diagram for a particular instance of a workflow process, so that users can get a graphical depiction of their work item status. The Workflow Monitor also displays a separate status summary for the work item, the process, and each activity in the process.

In Oracle Applications, you can also use the Oracle Workflow Manager component of Oracle Applications Manager as an additional administration tool for Oracle Workflow. Oracle Applications Manager is a tool that provides administrative and diagnostic capabilities for concurrent processing, Oracle Workflow, and other functionality in Oracle Applications. For more information, please refer to the Oracle Applications Manager online help.

Also, if you are using the standalone version of Oracle Workflow, you can use the standalone Oracle Workflow Manager component available through Oracle Enterprise Manager as an additional administration tool for Oracle Workflow. For more information, please refer to the Oracle Workflow Manager online help.

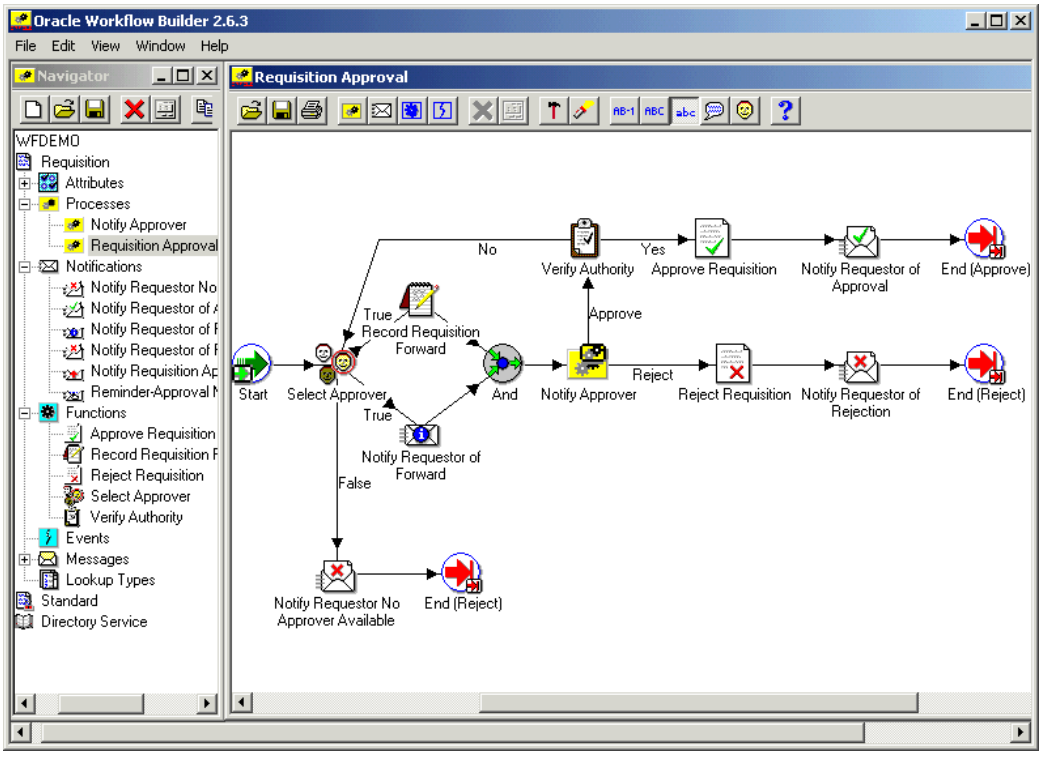
Workflow Processes

Oracle Workflow manages business processes according to rules that you define. The rules, which we call a workflow process definition, include the activities that occur in the process and the relationship between those activities. An activity in a process definition can be an automated function defined by a PL/SQL stored procedure or an external function, a notification to a user or role that may optionally request a response, a business event, or a subflow that itself is made up of a more granular set of activities.

A workflow process is initiated when an application calls a set of Oracle Workflow Engine APIs. The Workflow Engine takes over by driving the relevant work item defined by the application, through a specific workflow process definition. According to the workflow process definition, the Workflow Engine performs automated steps and invokes appropriate agents when external processing is required.

The following diagram depicts a simplified workflow process definition that routes a requisition to a manager or set of managers for approval.

Sample Workflow Process in Oracle Workflow Builder



We refer to the whole drawing as a process or process diagram. The icons represent activities, and the arrows represent the transitions between the activities. In the above example, new items are created for the process when a user creates and submits a requisition in the appropriate application.

This process contains several workflow activities implemented as PL/SQL stored procedures, including:

- Select Approver - to select, according to your business rules, who should approve the requisition.
- Verify Authority - to verify that a selected approver has the spending authority to approve the requisition.

2

Defining a Workflow Process

This chapter tells you how to use Oracle Workflow Builder to define a workflow process definition.

This chapter covers the following topics:

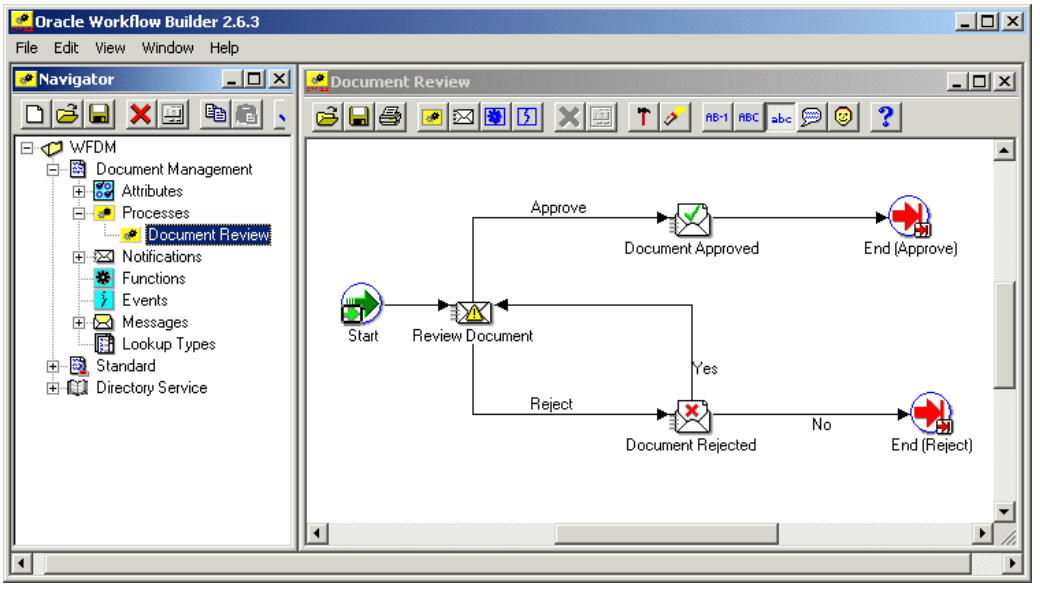
- Overview of Oracle Workflow Builder
- Viewing the Navigator Tree
- Creating Process Definitions in Oracle Workflow Builder
- Opening and Saving Item Types
- Quick Start Wizard Overview
- Using Oracle Workflow Builder with Different Server Versions
- Item Type Definition Web Page

Overview of Oracle Workflow Builder

Oracle Workflow Builder is a graphical tool for creating, viewing, and modifying workflow process definitions. It contains a Navigator window that you use to define the activities and components of your business process. You then assemble the activities in a process window to create a process diagram. See: Creating Process Definitions in Oracle Workflow Builder, page 2-5.

> **Note:** A workflow process definition can also be stored as a flat file, which can be opened and edited in a text editor so that the process definition can be spoken by a screen reader for greater user accessibility.

Oracle Workflow Builder Navigator Window and Process Window



Note: If you maximize the Navigator window or any process window in Oracle Workflow Builder, you will not be able to access the menu from your keyboard using the Alt key.

The Navigator Tree Structure

The Navigator window displays a navigator tree hierarchy for each data store that you open or load into Oracle Workflow Builder. A data store (primary branch) is a database connection or flat file that holds your workflow process definition. Within each data store there is at least one item type heading (secondary branch) that represents the grouping of a particular set of processes and its component objects. The following tertiary branches appear beneath each item type branch:

- Attributes - lists the attributes for the current item type. Item type attributes describe features of an item type. For example, if an item type is a purchase order requisition, then an item type attribute can be the requisition amount or the requisition ID. See: Item Type Attributes, page 3-2.
- Processes - lists the process activities or workflow process definitions for the current item type. See: Process Window, page 4-1 and Activities, page 3-52.
- Notifications - lists the notification activities associated with the current item type. A notification activity sends a message to a user or role. The message may prompt for a response or may simply provide information. See: Activities, page 3-52.
- Functions - lists the function activities associated with the current item type. A function activity represents a PL/SQL stored procedure that the Workflow Engine executes automatically. A function activity can also have activity attributes associated with it. See: Activities, page 3-52.
- Events - lists the event activities associated with the current item type. An event activity represents a business event that the process receives, raises, or sends. See: Activities, page 3-52.

- Messages - lists the messages that a notification activity associated with the current item type can send to a user or role. A message can have message attributes associated with it. See: Messages, page 3-22.
- Lookup Types - lists the lookup types associated with the current item type. A lookup type has one or more values called lookup codes associated with it. A lookup type is a list of values that can be referenced by a message, or by a notification, function, or process as its possible result type. See: Lookup Types, page 3-19.

  Note: Each data store also contains a Directory Service branch. The Directory Service branch lists all the directory service roles that you load from your Oracle Workflow database. See: Roles, page 4-19.

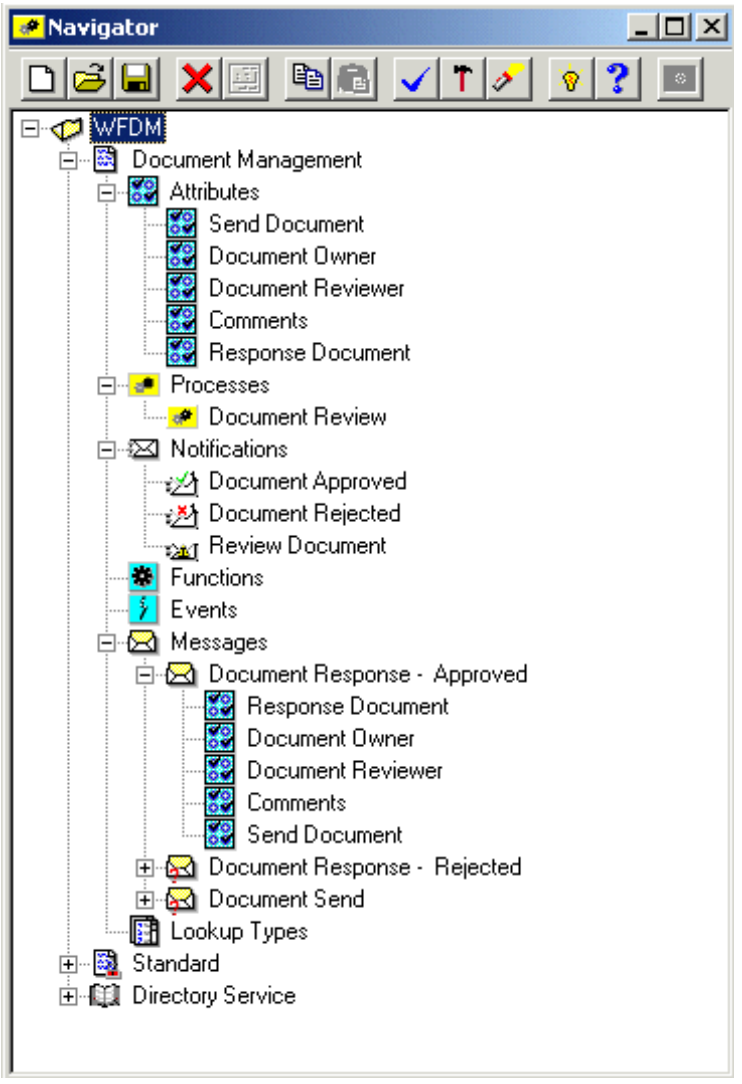
If the data store is a database connection and the database contains other item types that you have not loaded into Oracle Workflow Builder, a branch called Hidden Item Types appears. When you double-click on Hidden Item Types, you get a Show Item Types window that lets you load other item types into Oracle Workflow Builder.

Viewing the Navigator Tree

The navigator tree is organized much like the hierarchy of a file system, where you can expand branches that begin with a plus sign (+) to further sub-branches until you find your component of interest. Sub-branches appear indented below the branches from which they are expanded. Branches that are expanded are preceded by a minus sign (−). You can expand no further when a branch displays neither a plus nor minus sign. You can use either your mouse or the arrow keys on your keyboard to expand or collapse the navigator tree.

The Navigator window also contains a toolbar that you can use to perform actions within the Navigator window. See: Navigator Toolbar, page B-4.

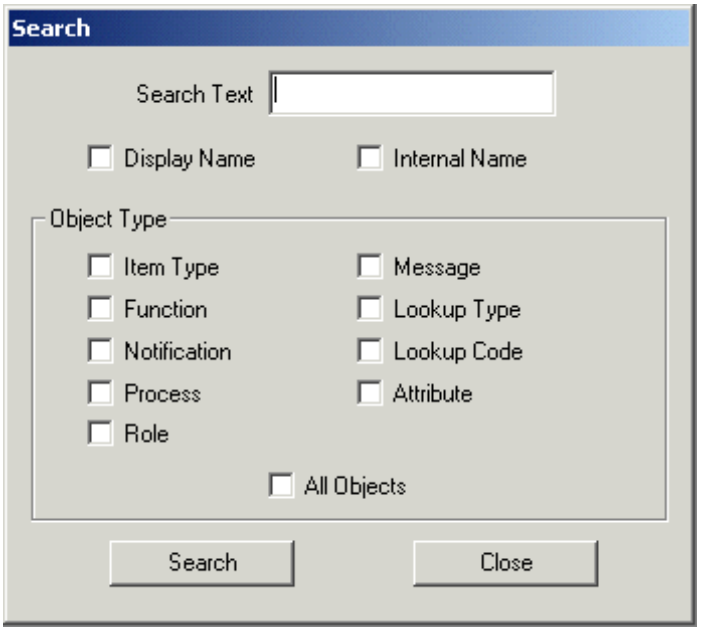
Navigator Window



To Find an Object in the Navigator Tree:

1. Choose Find... from the Edit menu to display a Search window that lets you specify search criteria to find an object in the navigator tree.

Search Window



2. Enter the text to search for in the Search Text field. The search is case insensitive and looks for the text pattern that you specify in the field that you specify.
3. Specify to search for this text in the object's Display Name or Internal Name.
4. Specify the object type to restrict this search to or check All Objects to search for the text within the property pages of all objects.
5. Choose Search.
6. You can choose Find Again from the Edit menu to repeat the search using the search criteria previously defined in the Search window.

Creating Process Definitions in Oracle Workflow Builder

Before using Oracle Workflow Builder, you should plan what your process needs to accomplish. In particular, determine what activities need to occur, the order of the activities, what results dictate the different branches of the process, who needs to be informed and what they need to know. Oracle Workflow provides several demonstration workflow examples. See: Sample Workflow Processes, page 10-1.

There are several ways you can go about creating a workflow process definition:

- Top-Down Design - If you prefer to approach your design from a high level, you can first sketch out the process diagram with activities, then go back later to create the supporting objects for each activity. See: To Create a Process Definition from Top-Down, page 2-8.
- Bottom-Up Design - If you prefer to take a more programmatic approach to your design, you can first define each of the supporting objects of your process before attempting to create a higher level process diagram. See: To Create a Process Definition From Bottom-Up, page 2-6.

Quick Start Wizard

The Quick Start Wizard helps you build a process definition from scratch using a process definition template. The Quick Start Wizard creates a new item type for your process, prompting you for the minimum required information. It then creates an outline process diagram from which you can flesh out with more activities. Once the Quick Start Wizard sets up the template, you can use either the top-down or bottom-up approach to complete the design. See: To Use the Quick Start Wizard, page 2-14.

Versioning and Dates of Effectivity

Oracle Workflow Builder assigns a version number to each new activity that you create. It also updates the version number whenever you make changes to an existing activity. It saves the new version of the activity to the database without overwriting older versions of the activity. In Oracle Workflow, activities also have dates of effectivity so that at any point in time, only one version of the activity is "in effect". If a process is running, Oracle Workflow uses the version of the activity that was in effect when the process was initiated. It does not switch versions of the activity mid-way through the process. Note that a process itself is an activity, so a process definition always remains constant until the process instance completes.

Oracle Workflow Builder also supports the concept of saving and loading process definitions according to an effective date. For example, you can load a definition into Oracle Workflow Builder that was effective at an earlier point in time. You can also save a definition to the database to be effective at some future time.

Note that Oracle Workflow Builder does not maintain version information for objects that are considered constant, such as item types, item type attributes, messages and lookup types. For these objects, their latest definition always apply, so you should always consider whether a change to any of these objects is backwards compatible. If the modification affects existing processes, you should create a new object rather than edit the existing object.

See: Modifying Objects in Oracle Workflow Builder, page 3-75.

Using the Edit Button in a Property Page:

To create an object in Oracle Workflow Builder, you enter information in the object's property page. Some of the information you provide can be selected from a list of values. If a poplist field yields values that are themselves defined from some other property pages in Oracle Workflow Builder, an Edit button appears to the right of that poplist. When you select a value from a poplist, you can choose the adjacent Edit button to display and edit the source property page(s) of the value. When you are done with the source property page(s) and choose OK or Cancel, you return to the original property page you were working on.

For example, if you create a notification activity, you must specify a Result Type for the activity. The Result Type poplist field lets you select the value `<None>` or some predefined lookup type. If you select a lookup type, you can then choose the adjacent Edit button to display the property page for that lookup type. When you finish viewing or editing the property page for that lookup type, you can choose OK or Cancel to return to the notification activity property page.

To Create a Process Definition From Bottom Up:

1. To start Oracle Workflow Builder, double-click the Oracle Workflow Builder icon located in the Application Development folder within the `Oracle - <SID NAME>`

program group. If you are using Windows 98 or NT 4.0 or higher, you can also select the Oracle Workflow Builder icon from the appropriate program folder of the Start menu.

2. Choose New from the File menu to create a workspace for your new process definition.

   Tip: Alternatively, you can use the Quick Start Wizard to first create the framework for your new process definition. Once the Quick Start Wizard creates your new item type and new process activity, you can skip to step 4 below to begin defining the supporting objects for the new item type and process activity. See: To Use the Quick Start Wizard, page 2-14.

3. Create a new item type. The item type classifies the work item to be managed by the process. See: To Create an Item Type, page 3-6.

4. You can define item type attributes to fully describe your item type and have the activities in your process refer to these attributes for information. See: To Define an Item Type or Activity Attribute, page 3-7.

5. Create new lookup types. See: To Create Lookup Types, page 3-20.

   Before defining an activity, you should define the lookup type that represents your activity's Result Type. A Result Type is a list of possible results that an activity can have upon completion. After defining a lookup type and an activity, you can drag the lookup onto an activity in the navigator tree to assign that lookup as the activity's result type. Lookup types can also be referenced by item type attributes, activity attributes, messages, or message attributes.

6. Create new messages. See: To Create a Message, page 3-42.

   If you wish to create a notification activity for your process, you should first create the message that you want the notification activity to send. You can drag a new message onto a notification activity in the navigator tree to assign the message to that activity.

   You can also create message attributes for the message. You can incorporate message attributes of type 'Send' into a message that are token substituted at runtime to provide dynamic content. You can also define message attributes of type 'Respond' to prompt the notification recipient for a response. See: To Define a Message Attribute, page 3-46.

7. Create a new process activity, page 3-66, notification activity, page 3-58, function activity, page 3-60, or event activity, page 3-62. You may also use predefined standard activities associated with the Standard item type. See: Activities, page 3-52 and Standard Activities, page 5-1.

   You need to define at least one process activity that represents your high level process diagram. The process diagram establishes the relationship of all the activities in your process.

8. Diagram the process.

   Display the Process window for your process activity to diagram the activities and transitions that define your workflow process. You can drag activities from the navigator tree into the Process window. See: Diagramming a Process, page 4-3.

9. Save your work by choosing Save or Save As from the File menu. See: To Save Your Work, page 2-11.
10. In a database accessible by your Oracle Workflow server, create the PL/SQL stored procedures called by your PL/SQL function activities. You can do this through SQL*Plus or the Oracle Procedure Builder. See: Oracle Workflow Procedures and Functions, *Oracle Workflow API Reference* and Standard API for PL/SQL Procedures Called by Function Activities, page 6-2.

To Create a Process Definition from Top Down:

1. To start Oracle Workflow Builder, double-click on the Oracle Workflow Builder icon located in the Application Development folder within the `Oracle - <SID NAME>` program group. If you are using Windows 98 or NT 4.0 or higher, you can also select the Oracle Workflow Builder icon from the appropriate program folder of the Start menu.
2. Use the Quick Start Wizard to create the framework for your new process definition. Specify the requested information for the new item type and new process activity. See: To Use the Quick Start Wizard, page 2-14.
3. A Process window appears, that shows a Start and an End activity node. Create your process diagram by defining new activity nodes to place between the Start and End nodes. See: To Define Nodes in a Process, page 4-5.

   You may also use predefined standard activities associated with the Standard item type. See: Standard Activities, page 5-1.
4. Model your process by drawing transitions between your activities. See: Diagramming a Process, page 4-3.
5. Save your work by choosing Save or Save As from the File menu. See: To Save Your Work, page 2-11.

   > **Important:** When you save your work, Oracle Workflow automatically validates the process definition for any invalid or missing information and displays what it finds in a Workflow Error verification window. The Workflow Error window is non-modal, so you can keep it up on your screen while you go back to your process to correct the problems that are identified. You can also save your work as is, and fix the problems later. Use the Copy button to copy the information to the clipboard if you want to paste it into another document for later reference. If you save your work without correcting the problems, the Workflow Error window will appear when you open this process definition later.

To Modify a Process Definition:

1. To start Oracle Workflow Builder, double-click on the Oracle Workflow Builder icon located in the Application Development folder within the `Oracle - <SID NAME>` program group. If you are using Windows 98 or NT 4.0 or higher, you can also select the Oracle Workflow Builder icon from the appropriate program folder of the Start menu.
2. Choose Open from the File menu to open a connection to the database or file that contains the process definition you want to modify. See: To Access Process Definitions in an Existing Data Store, page 2-9.

3. Select and expand the existing item type associated with the process definition you want to modify.
4. You can modify an item type, item type attribute, lookup, message, message attribute, process activity, notification activity, function activity, or activity attribute. See: To Create an Item Type, page 3-6, To Define an Item Type or Activity Attribute, page 3-7, To Create Lookup Types, page 3-20, To Create a Message, page 3-42, To Define a Message Attribute, page 3-46, or Activities, page 3-52.
5. You can also modify the process diagram by displaying the Process window for your process activity. See: Diagramming a Process, page 4-3.
6. Save your work by choosing Save or Save As from the File menu. See: To Save Your Work, page 2-11.

Related Topics

Deleting Objects in Oracle Workflow Builder, page 3-74

Modifying Objects in Oracle Workflow Builder, page 3-75

Item Type Definition Web Page, page 2-18

Using Oracle Workflow Builder with Different Server Versions, page 2-15

Opening and Saving Item Types

All processes are associated with an item type. An item type can include one or more processes. You can save an item type to a database or to a flat file. When you save your work to a database, you actually save everything in the current data store that has been modified. When you save your work to a flat file, you actually save everything in the current data store to the file. You can also load an item type into Oracle Workflow Builder from a database or flat file. Opening an item type automatically retrieves all the attributes, messages, lookups, notifications, functions and processes associated with that item type.

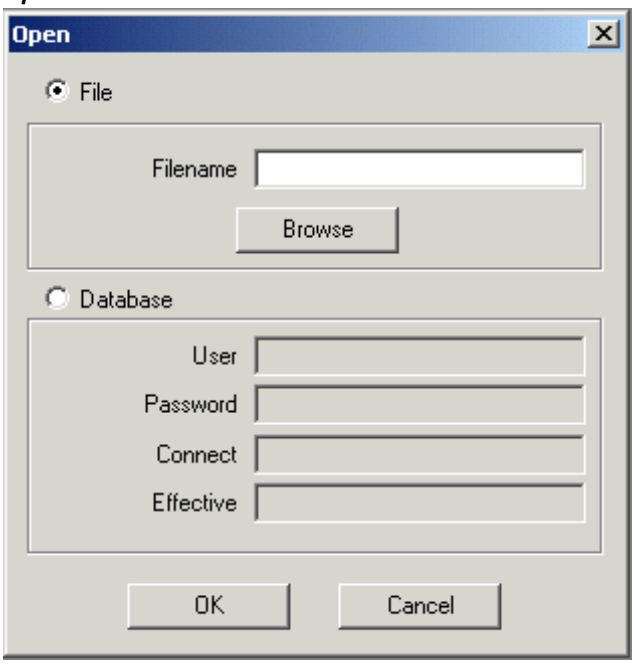
> **Important:** Always save a copy of your workflow process definition as a flat file and check that file into a source control system to maintain a working version of your process definition. Avoid using the process definition stored in your database as your source controlled version, as others with access to the database can update the definition.

> **Note:** To connect from Oracle Workflow Builder to a database, the language of your Oracle Workflow Builder installation must match one of the available languages of the Oracle Workflow Server installation in the database.

To Access Process Definitions in an Existing Data Store:

1. To start Oracle Workflow Builder, double-click on the Oracle Workflow Builder icon located in the Application Development folder within the `Oracle - <SID NAME>` program group. If you are using Windows 98 or NT 4.0 or higher, you can also select the Oracle Workflow Builder icon from the appropriate program folder from the Start menu. In Oracle Workflow Builder, select Open... from the File menu.

Open Window



2. Select database or file to connect to the source containing the item type to which your process definition is associated.
3. To open a file: Provide the complete file path and choose OK, or use Browse to locate and open the file (extension `.wft`).

   Note: You can also drag and drop a `.wft` file from the Microsoft Windows 98/2000/XP/NT 4.0 Explorer or Microsoft Windows NT File Manager into the navigator tree to open that file in Oracle Workflow Builder.

   Note: When you use Browse to find and open a file, the current directory that you open the file from becomes the new default directory from which you open files in the future. This default directory persists until you use Browse again to locate another file.

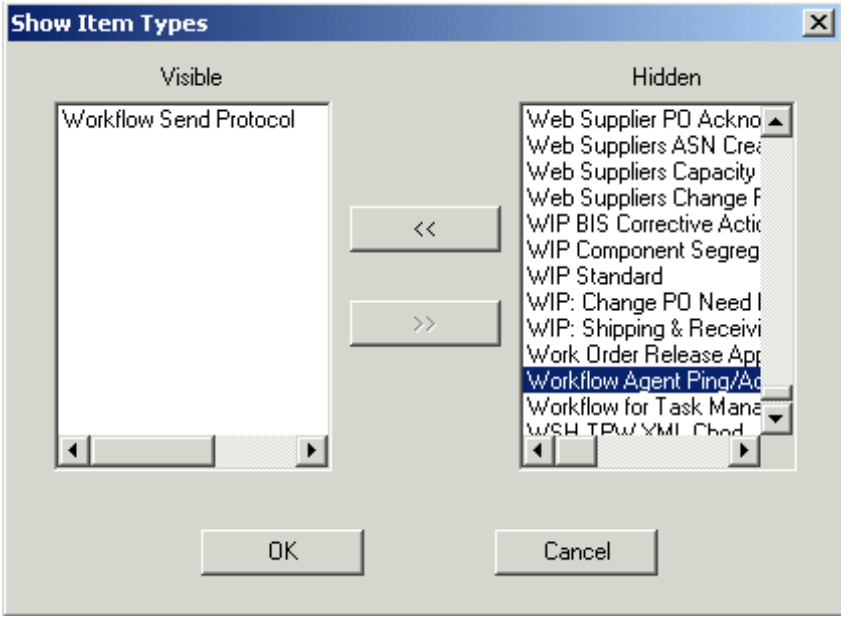
4. To open a database connection: Enter the username and password for the database. Enter the name of the database alias or connect string and choose OK.

   Note: If you are using the version of Oracle Workflow embedded in Oracle Applications, use the `APPS` schema to connect to the database.

5. If you wish to retrieve a process definition that was effective at a particular point in time, you can specify a date and time in the Effective field and have Oracle Workflow Builder retrieve that data from the database. The format that you should use to specify the date and time depends on the date and time preferences defined in the Regional Settings of your Windows Control Panel.
6. If multiple item types exist in the data store, the Show Item Types window appears. Select from the Hidden list, the item type(s) you want to view, and choose

<< to move it into the Visible list. Choose OK to load these item types into the navigator tree.

Show Item Types Window



7. If at any time you want to view and modify item types that are hidden in the current data store, you can double-click on the Hidden Item Types branch in the navigator tree to display the Show Item Types window and select the item types you want to show. You can also choose Show/Hide Item Types from the File menu to display the Show Item Types window.

   Note: You can copy item types from one store to another in any order even if the item types reference each other. However, you may get validation errors due to foreign key references. Pay attention to these errors as they may indicate that you need to also copy other item types into the new store to resolve the foreign key references. The final process definition in the new store will be valid as long as all referenced item types are copied to the new destination store.

8. When you finish working, choose Save from the File menu to preserve your changes and make them effective immediately. See: To Save Your Work, page 2-11.

To Save Your Work:

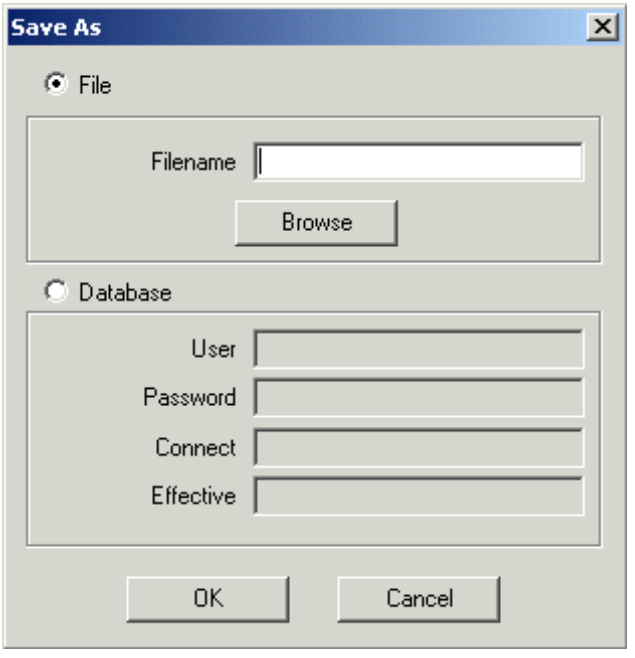
1. Choose Save from the File menu to save your work and make the changes immediately effective.

   When you use the Save command, you save all modified objects in the currently selected data store (even those that are hidden) back to that data store. If you want to save only specific item types, then you must create a new data store, and copy the specific item types you want to save into the new store and save the new store.

   Important: Oracle Workflow Builder can save your work to the database using one of two modes. In the "About Oracle Workflow

Builder" dialog box from the Help menu, there is a check box called "Allow modifications of customized objects". If you check this check box, Oracle Workflow Builder saves your edits in 'upload' mode, overwriting any protected objects that you have access to modify, as well as any previously customized objects. If you uncheck this check box, Oracle Workflow Builder runs in 'upgrade' mode and will only save edits to protected objects that you have access to change and will not overwrite objects that have been previously customized. These two modes match the upgrade and upload behavior of the Workflow Definitions Loader program. As the default, the check box is unchecked. See: To Set the Access Level for an Object, page 3-18 and Using the Workflow Definitions Loader, *Oracle Workflow Administrator's Guide*.

Save As Window



2. If you want to save your work to a different data store (database or flat file), or if you want to save it to a database with an effective date other than the current system date, then choose Save As... from the File menu. Use the Save As window to specify the file or database you want to save your process definition to, and the date when you want your process definition to take effect in the database. You can leave the Effective field blank to save and make the changes effective immediately. See: Version/Effective Date, *Oracle Workflow API Reference*.

   Note: If you save your work to a database with a future effective date, and then in the same Oracle Workflow Builder session, continue to modify your process and later choose Save from the File menu, you automatically save the process definition to the same database using the previously specified effective date.

3. Note that when you save your work, Oracle Workflow automatically validates the process definition for any invalid or missing information and displays what it finds in

a Workflow Error verification window. You can either correct the information before saving your work, or save your work as is, and fix the problems later. Use the Copy button to copy the information from the Workflow Error window to the clipboard for later reference. If you save your work without correcting the problems, the Workflow Error window will reappear when you reopen your process definition.

4. Choose Close Store from the File menu to close your connection to the current database or file data store.
5. Choose Exit from the File menu to exit Oracle Workflow Builder.

> **Important:** The Close Store and Exit options from the File menu are enabled only when the Navigator window is the current window.

To Start Oracle Workflow Builder from the MS-DOS Prompt:

Rather than starting Oracle Workflow Builder by double-clicking on its Windows icon, you can also type in a command at the MS-DOS prompt and specify the file or database to connect to.

1. In an MS-DOS prompt window, type the following command to start Oracle Workflow Builder with a specific workflow data file, where *<filename.wft>* represents the full path and name of the data file:

   ```
   wfbldr <filename.wft>
   ```

2. To start Oracle Workflow Builder with a specific database connection, type the following command at the MS-DOS prompt, where *<username/password@connect>* represents the database account information to connect to:

   ```
   wfbldr -c <username/password@connect>
   ```

   > **Note:** If you run Oracle Workflow Builder in Microsoft Windows 98 or Windows NT 4.0 or higher, you can also double-click on a workflow data file (`.wft`) from the Windows Explorer to automatically open that file and start Oracle Workflow Builder.

   > **Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications, use the `APPS` schema to connect to the database.

3. To start Oracle Workflow Builder and open a specified item type in a data store, append the following to the appropriate command shown in Step 1 or 2, where *<item_type>* represents the internal name of the item type you want to open:

   ```
   -E <item_type>
   ```

   For example:

   ```
   wfbldr wfdemo.wft -E wfdemo
   ```

4. To start Oracle Workflow Builder and open a specified process diagram in a data store, append the following to the appropriate command shown in Step 1 or 2, where *<item_type:process>* represents the internal names of the item type and process you want to open:

   ```
   -E <item_type:process>
   ```

   For example:

   ```
   wfbldr wfdemo.wft -E WFDEMO:NOTIFYAPPROVER
   ```

Related Topics

Using the Workflow Definitions Loader, *Oracle Workflow Administrator's Guide*

Creating a Shortcut to a Workflow Process, page 4-18

Item Type Definition Web Page, page 2-18

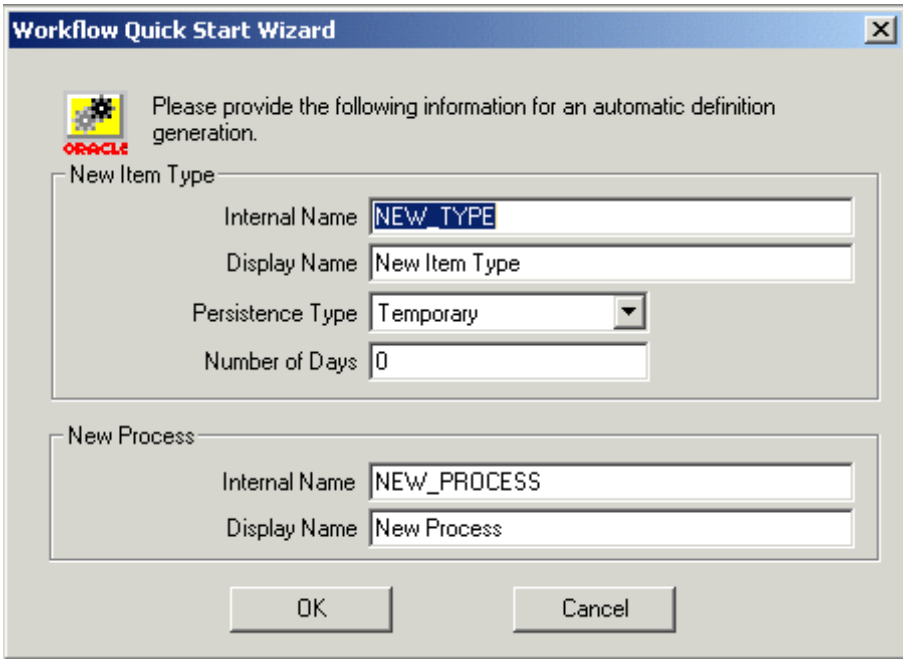
Quick Start Wizard Overview

The Quick Start Wizard lets you begin designing a workflow process immediately. It first loads a file called `wftemplate.wft` that is an outline of all the mandatory objects you need to build a workflow process and then displays a Process window for you to diagram your process. Once you initiate the Quick Start Wizard, you can take the bottom-up or top-down approach to complete your workflow process definition.

To Use the Quick Start Wizard:

1. Select Quick Start Wizard from the File menu.

Workflow Quick Start Wizard Window



2. The Workflow Quick Start Wizard window prompts you for the following mandatory information:
   - New Item Type
     - Internal Name - Specify an all uppercase internal name with a maximum of eight characters. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying an item type.

       Important: To update the internal name of an item type once it is defined, you must use a special SQL script called `wfchitt.sql`. See: Wfchitt.sql, *Oracle Workflow Administrator's Guide*.

Caution: Do not include colons "`:`" or leading/trailing spaces in your internal name.

- Display Name - Enter a translatable Display Name for the item type.
- Persistence Type - Specify Temporary or Permanent persistence for the status audit trail of the item type.
- Days - If Persistence Type is Temporary, specify the number of days from the time an item type instance completes before its status audit trail can be purged. See: Persistence Type, page 3-4.

- New Process
  - Internal Name - Specify an all uppercase internal name.

    Important: To update the internal name of an activity once it is defined, you must use a special SQL script called `wfchact.sql`. See: Wfchact.sql, *Oracle Workflow Administrator's Guide*.

    Caution: Do not include colons "`:`" or leading/trailing spaces in your internal name.

  - Display Name - Enter a translatable Display Name for the process activity. The Display Name also appears in the title bar of your Process window.

3. The Quick Start Wizard does the following:
   - Creates a new data store called "`Untitled-`*`n`*" in the Navigator window.
   - Uses the information you entered in the Workflow Quick Start Wizard window to create a new item type and process activity in the data store.
   - Loads the Standard item type into the new data store so that you can include standard activities in the process you create.
   - Opens the Process window for the new process activity you defined. The Process window displays a Start and an End activity.
4. You can now customize your process definition in one of two ways:
   - Take a bottom-up design approach by first creating activities and all their supporting objects before trying to draw a workflow diagram. See: To Create a Process Definition From Bottom-Up, page 2-6.
   - Take a top-down design approach by creating activities that contain minimum information so you can draw the workflow diagram first. You can go back later to fill in the details of each activity and its supporting objects. See: To Create a Process Definition from Top-Down, page 2-8.

Using Oracle Workflow Builder with Different Server Versions

The Oracle Workflow Builder Release 2.6.3 shipped with Oracle Applications is compatible with all versions of the Oracle Workflow server embedded in Oracle Applications Release 11*i*. The Oracle Workflow Builder Release 2.6.3.5 available from the Oracle Technology Network (http://www.oracle.com/technology/) is compatible with

Release 2.6.4, Release 2.6.3.5, Release 2.6.3, Release 2.6.2, Release 2.6.1, base Release 2.6, and Release 2.5 of the standalone version of the Oracle Workflow server.

> **Note:** For Release 2.6.4, you can install the Oracle Workflow Builder from the same Oracle Technology Network client download that was originally made available for Oracle Workflow Release 2.6.3.5. There are no changes in the Oracle Workflow Builder for Release 2.6.4. Ensure that you install the Oracle Workflow Builder into a new Oracle home. You can then use this version of the Oracle Workflow Builder to connect to an Oracle Workflow Release 2.6.4 server.

- You can create, view, and modify workflow process definitions that include new features introduced in Release 2.6, such as Business Event System components and, for standalone Oracle Workflow only, external Java function activities. The current Oracle Workflow Builder versions can upload and download these process definitions to a database with Oracle Workflow Server Release 2.6 installed.
- Alternatively, if you do not want to take advantage of the Release 2.6 features, you can create, view, and modify workflow processes that include only Release 2.5 features. The current Oracle Workflow Builder versions can upload and download these process definitions to a database with Oracle Workflow Server Release 2.5 installed.

Using the Release 2.6.3 Oracle Workflow Builder with a Release 2.6 Embedded Server, or the Release 2.6.3.5 Oracle Workflow Builder with a Release 2.6.4, 2.6.3.5, 2.6.3, 2.6.2, or 2.6.1 Standalone Server:

If you are using the Oracle Workflow Builder with Oracle Workflow Server Release 2.6 embedded in Oracle Applications, or with the standalone version of the Oracle Workflow Server Release 2.6.4, Release 2.6.3.5, Release 2.6.3, Release 2.6.2, or Release 2.6.1, you can use all currently available features for your version of Oracle Workflow, either standalone or embedded, in your workflow processes. You can save these process definitions to the database and open process definitions from the database to view or modify them.

You can also open existing process definitions that were created with a Release 2.5 or earlier Release 2.6 Oracle Workflow Builder and save these process definitions to a database with Oracle Workflow Server Release 2.6 embedded in Oracle Applications, or with standalone Oracle Workflow Server Release 2.6.4.

Using the Release 2.6.3.5 Oracle Workflow Builder with a Release 2.6 Standalone Server:

If you are using the Oracle Workflow Builder with the base standalone version of Oracle Workflow Server Release 2.6, you can include most of the currently available features in your workflow processes. However, you must not use the Event Parameter lookup code which is now available in the Event Property lookup type, because this feature was added in the version of Release 2.6 embedded in Oracle Applications. Do not use the Event Parameter lookup code in any custom activities, and do not select the Event Parameter property in any of the following standard Workflow activities:

- Get Event Property
- Set Event Property
- Compare Event Property

You can open existing process definitions that were created with the base Release 2.6 Oracle Workflow Builder, view or modify these process definitions using only the base Release 2.6 components, and save the definitions to a database with base standalone Oracle Workflow Server Release 2.6.

You can also create new workflow processes using only base Release 2.6 components. However, the version of the Standard item type used by the Oracle Workflow Builder Release 2.6.3.5 contains some later Release 2.6 components. If you want to save a new process definition to a database with base standalone Oracle Workflow Server Release 2.6, perform the following steps:

1. Create a new workflow process definition.
2. Force delete the Standard item type from your data store.
3. Force save the process definition to the database with base standalone Oracle Workflow Server Release 2.6.
4. Reopen the process definition from the database. The process definition now includes the base Release 2.6 version of the Standard item type.
5. Continue defining your workflow process using only base Release 2.6 components.

Using the Release 2.6.3.5 or Release 2.6.3 Oracle Workflow Builder with a Release 2.5 Standalone or Embedded Server:

If you are using the Oracle Workflow Builder with either the standalone or the embedded version of Oracle Workflow Server Release 2.5, you must include only Release 2.5 features in your workflow processes. You must not use any of the following new features introduced in Release 2.6:

- Event activities
- Item attributes of type Event
- External Java function activities

You can open existing process definitions that were created with the Release 2.5 Oracle Workflow Builder, view or modify these process definitions using only Release 2.5 components, and save the definitions to a database with Oracle Workflow Server Release 2.5.

You can also create new workflow processes using only Release 2.5 components. However, the version of the Standard item type used by the Oracle Workflow Builder Release 2.6.3 or Release 2.6.3.5 contains some Release 2.6 components. If you want to save a new process definition to a database with Oracle Workflow Server Release 2.5, perform the following steps:

1. Create your workflow process definition.
2. Force delete the Standard item type from your data store.
3. Force save the process definition to the database with Oracle Workflow Server Release 2.5.
4. Reopen the process definition from the database. The process definition now includes the Release 2.5 version of the Standard item type.

Item Type Definition Web Page

In standalone Oracle Workflow, the Web-based Item Type Definition page provides you with distributed access to workflow definitions stored in your Oracle Workflow database. The page provides a detailed view of the attributes, processes, notifications, functions, events, messages, and lookup types that are associated with a given item type, allowing you to present or do a design review of your workflow process.

To display an item type definition, you use the Find Item Type Web page to first query for an item type. You can query for an item type based on an effective date and time.

The Item Type Definition page then appears. The information is displayed in two frames, modeled like the Oracle Workflow Builder, so that you can review the contents easily and effectively. The left frame lists all the objects in your item type definition in an expandable navigator tree. The right frame displays the details of the object you select in the navigator tree. You can also select either frame at any time and use your Web browser to print all the information in that frame.

To Query an Item Type:

1. Enter the following URL in your Web browser:

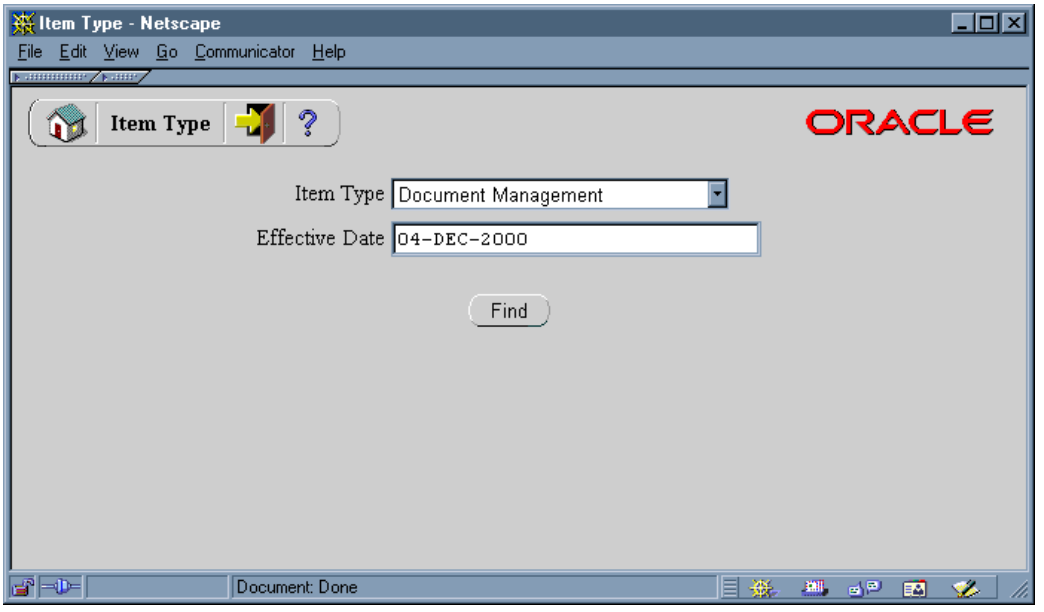
   ```
   <webagent>/wf_item_definition.find_item_type
   ```

   Replace `<webagent>` with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

   Important: This URL accesses a secured page, so if you have not yet logged on as valid user in the current Web session, you will be prompted to do so before the page appears.

   Note: You can also access the Find Item Type Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

Item Type Page



2. Use the Item Type poplist field to select an item type.
3. Specify the effective date and time of the item type definition you want to display using the format specified in the Date Format field of your User Preferences Web page. See: Setting User Preferences, *Oracle Workflow User's Guide*.
4. Choose Find to display the item type in the Item Type Definition Web page.

To Review an Item Type Definition:

1. The Item Type Definition Web page displays two frames. The frame on the left lists the components of an item type definition in hierarchical format similar to the navigator tree in Oracle Workflow Builder. The frame on the right lists the details of each component.
2. Click on any component link in the left hand frame to display the details of that component in the right hand frame.

3

Defining Workflow Process Components

This chapter tells you how to use Oracle Workflow Builder to define the components necessary to compose a workflow process diagram.

This chapter covers the following topics:

- Workflow Process Components
- Item Types
- Attributes
- Defining Item Types and Attributes
- Allowing Access to an Object
- Lookup Types
- Defining Lookup Types
- Messages
- Defining Messages
- Activities
- Defining Activities
- Voting Activity
- Example Voting Methods
- Deleting Objects in Oracle Workflow Builder
- Modifying Objects in Oracle Workflow Builder

Workflow Process Components

Depending on the workflow process you wish to create, you need to define all or some of the following types of components to make up the process:

- Item Types, page 3-2
- Attributes, page 3-5
- Lookup Types, page 3-19
- Messages, page 3-22
- Activities, page 3-52
- Roles, page 4-19

Note: When defining internal names for workflow objects in Oracle Workflow Builder, use only characters from the ASCII character set.

Item Types

An item type is a classification of the components that make up a workflow process. You must associate any component that you create for a process, such as a function activity or a message, with a particular item type. Often it makes sense to define an item type so that it describes the item being managed by your workflow process. For example, purchase order requisition can be an item type while a purchase order requisition identified by a particular ID number is an item of that item type. See: To Create an Item Type, page 3-6.

Item Type Attributes

An item type attribute is a property associated with a given item type. It acts as a global variable that can be referenced or updated by any activity within a process. An item type attribute often provides information about an item that is necessary for the workflow process to complete. For example, the "Workflow Demonstration" item type has an item type attribute called "Requisition Amount." An activity in our example Requisition Approval process requires the value of this item type attribute to determine if a selected approver has the authority to approve a requisition of that amount.

Applications as well as function activities can reference and set item type attributes using the Oracle Workflow Engine APIs. You can define and maintain as many item type attributes as necessary for an item type. You should define as an item type attribute, any information that will be required by an activity in your process, or any information that will need to be sent in a notification message. See: To Define a Message Attribute, page 3-46 and Item Attributes, *Oracle Workflow Administrator's Guide*.

Attribute Types

There are ten attribute types, as shown below. The type determines what values are acceptable and how the attribute is used.

- Text - The attribute value is a string of text.

  Note: Due to security considerations, Oracle Workflow does not permit HTML content to be passed in attributes of type text. If Oracle Workflow encounters HTML tags in a text attribute, escape characters will be applied to display the content as plain text rather than executing the HTML. If you want to apply HTML formatting to your content, define an attribute of type document instead, and pass your HTML content as either a PL/SQL document or PL/SQL CLOB document.

- Number - The attribute value is a number with the optional format mask you specify.
- Date - The attribute value is a date with the optional format mask you specify.
- Lookup - The attribute value is one of the lookup code values in a specified lookup type.
- Form - The attribute value is an Oracle Applications internal form function name and its optional form function parameters. This attribute type is not relevant for the standalone version of Oracle Workflow.

If you include a form-type attribute in a notification message as a message attribute, the notification, when viewed from the Notification Details Web page, displays an attached form icon that lets users drill down to the referenced form. See: Overview of Menus and Function Security, *Oracle Applications Developer's Guide*.

- URL - The attribute value is a Universal Resource Locator (URL) to a network location. If you reference a URL attribute in a notification message as a message attribute, the notification, when viewed from the Notification Details Web page or as an HTML-formatted e-mail, displays a link to the URL specified by the URL attribute. The user can complete an activity or see additional information related to the activity by accessing that URL. You can also use a URL attribute to display an inline image in a notification message, when the notification is viewed from the Notification Details Web page or as an HTML-formatted e-mail.
- Document - The attribute value is an attached document. You can specify the following types of documents in the default value field:
  - PL/SQL document - A document representing data from the database as a character string, generated from a PL/SQL procedure.
  - PL/SQL CLOB document - A document representing data from the database as a character large object (CLOB), generated from a PL/SQL procedure. The CLOB can contain plain text, HTML, an Adobe Acrobat Portable Document Format (PDF) document, a Microsoft Rich Text Format (RTF) document, or, if your database version is Oracle9*i* Database or higher, binary data encoded to base64.
  - PL/SQL BLOB document - A document representing data from the database as a binary large object (BLOB), generated from a PL/SQL procedure. The BLOB can contain an image or other types of application files that are stored as binary data.
  - Oracle Applications Framework region - In Oracle Applications, a JSP call to an Oracle Applications Framework region for inclusion in a notification message

  See: To Define a Document Attribute, page 3-13.
- Role - The attribute value is the internal name of a role. If a message attribute of type role is included in a notification message, the attribute automatically resolves to the role's display name, eliminating the need for you to maintain separate attributes for the role's internal and display names. Also when you view the notification from a Web browser, the role display name is a hypertext link to the e-mail address for that role. To set a default value for the attribute, you must initially load roles from the database. See: Roles, page 4-19.
- Attribute - The attribute value is the internal name of another existing item type attribute that you want to maintain references to in a process.
- Event - The attribute value is a Business Event System event message in the standard `WF_EVENT_T` structure. See: Event Message Structure, *Oracle Workflow API Reference*.

  > **Note:** If you store an event message in an item attribute of type event, you can access the event data within that event message by creating an item attribute of type URL and setting the value of the URL attribute to reference the event data. See: SetItemAttribute, *Oracle Workflow API Reference*.

Persistence Type

When you define an item type, you must also specify its persistence type. The persistence type controls how long a status audit trail is maintained for each instance of the item type. If you set Persistence to Permanent, the runtime status information is maintained indefinitely until you specifically purge the information by calling the procedure *WF_PURGE.TotalPerm()*.

If you set an item type's Persistence to Temporary, you must also specify the number of days of persistence ('*n*'). The status audit trail for each instance of a Temporary item type is maintained for at least '*n*' days of persistence after its completion date. After the '*n*' days of persistence, you can then use any of the WF_PURGE APIs to purge the item type's runtime status information. See: WF_PURGE, *Oracle Workflow API Reference*.

If you set an item type's Persistence to Synchronous, Oracle Workflow expects instances of that item type to be run as forced synchronous processes with an item key of `#SYNCH`. Forced synchronous processes complete in a single SQL session from start to finish and never insert into or update any database tables. Since no runtime status information is maintained, you do not normally need to perform any purging for a process with the Synchronous persistence type. However, if you run the process with a unique item key in asynchronous mode for testing or debugging purposes, Oracle Workflow does maintain runtime status information for that process instance. You can purge this information by changing the item type's Persistence to Temporary and running any of the WF_PURGE APIs. Then change the item type's Persistence back to Synchronous. See: Synchronous, Asynchronous, and Forced Synchronous Processes, *Oracle Workflow API Reference*, WF_PURGE, *Oracle Workflow API Reference*, and Purging for Performance, *Oracle Workflow Administrator's Guide*.

> **Note:** In Oracle Applications, you may also use the Purge Obsolete Workflow Runtime Data concurrent program to purge obsolete item type runtime status information. The executable name for this concurrent program is "Oracle Workflow Purge Obsolete Data" and its short name is FNDWFPR. See: Purge Obsolete Workflow Runtime Data, *Oracle Workflow API Reference*.

Item Type Selector Function

If your item type has or will have more than one runnable process activity associated with it, define a PL/SQL function that determines which process activity to run in a particular situation. For example, you may have two different requisition approval process activities associated with the same item type. The process that Oracle Workflow executes may vary depending on how and where the requisition originates. Your selector function would determine which process would be appropriate in any given situation.

You can also extend the Selector function to be a general callback function so that item type context information can be reset as needed if the SQL session is interrupted during the execution of a process. This is particularly important in the Oracle Applications scenario when you view a notification from the Notification Details Web page and attempt to launch a form that is associated with the notification. Oracle Workflow calls the selector/callback function for your item type in '`TEST_CTX`' mode to test the Oracle Applications context before turning the form launch over to the Oracle Application Object Library function security system. In '`TEST_CTX`' mode, the selector/callback function can perform whatever logic necessary to determine whether it is appropriate to launch the form. See: Standard API for an Item Type Selector or Callback Function, page 6-10.

External Document Integration

Documents have an enormous impact in the operations of an organization. With the explosion of digital media and the worldwide Web, electronic documents of a wide variety of formats, including non-printed media, are forcing organizations to address the management of these documents. The value of information in these documents can be maintained only if the documents can be managed and shared.

In a workflow process, you can attach documents generated by a PL/SQL procedure, which we call PL/SQL, PL/SQL CLOB, or PL/SQL BLOB documents. You attach a document to a workflow process by referencing the document in a predefined item attribute or message attribute of type Document. See: Attribute Types, page 3-2, To Define an Item Type or Activity Attribute, page 3-7, and To Define a Message Attribute, page 3-46.

For PL/SQL, PL/SQL CLOB, or PL/SQL BLOB documents, the item or message attribute's value would be the name of the PL/SQL package and procedure used to generate the document. The PL/SQL procedure must follow an Oracle Workflow standard interface. See: Standard APIs for "PL/SQL" Documents, page 6-13.

For PL/SQL and PL/SQL CLOB documents that contain text or HTML, the document generated by the PL/SQL procedure can either be displayed within the text of a notification or included as an attachment. Other types of content in PL/SQL CLOB documents, as well as all PL/SQL BLOB documents, can only be included as attachments.

- In addition to text or HTML, PL/SQL CLOB documents that are included as attachments can contain PDF or RTF documents or, if your database version is Oracle9*i* Database or higher, binary data encoded to base64.
- PL/SQL BLOB documents that are included as attachments can contain images or other application files that are stored as binary data.

In Oracle Applications, Oracle Workflow also supports using attributes of type document to embed an Oracle Applications Framework region in a notification. See: Embedding Oracle Applications Framework Regions in Messages, page 3-38.

> **Note:** An Oracle Applications Framework region can only be displayed within the message body of a notification. It cannot be included as an attachment to the notification.

Attributes

You can create an attribute for any of the following workflow process components:

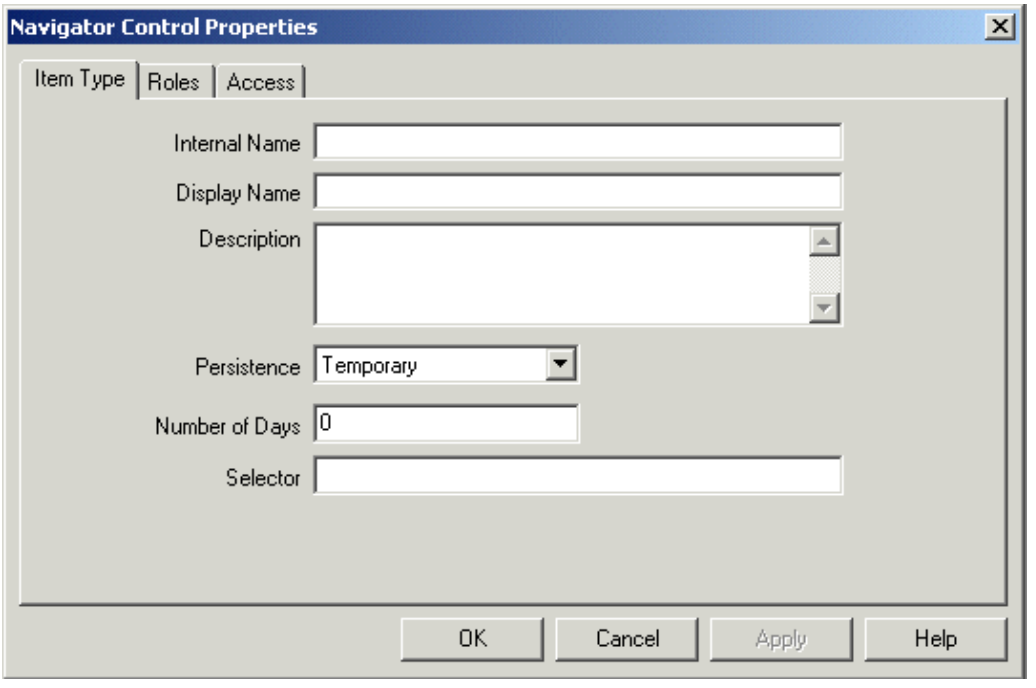
- Item type. See: To Define an Item Type or Activity Attribute, page 3-7.
- Function activity. See: To Define an Item Type or Activity Attribute, page 3-7.
- Event activity. See: To Define an Item Type or Activity Attribute, page 3-7.
- Notification activity. See: To Define an Item Type or Activity Attribute, page 3-7.
- Message. See: To Define a Message Attribute, page 3-46.

Defining Item Types and Attributes

To Create an Item Type:

1. Open an existing data store or, select New from the File menu to create a new data store to define this new item type. Then define a new item type in the navigator tree by choosing New Item Type from the Edit menu. An Item Type property page appears.

Item Type Property Page



2. Every item type has an all-uppercase internal name, which is a maximum of eight characters long. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying an item type.

   Important: To update the internal name for an item type once it is defined, you must use a special SQL script called `wfchitt.sql`. You should only use this script to correct errors in an item type's internal name during design time. Do not use this script to rename item types that are involved in running instances of processes. See: Wfchitt.sql, *Oracle Workflow Administrator's Guide*.

   Caution: Do not include colons "`:`" or leading/trailing spaces in your internal name.

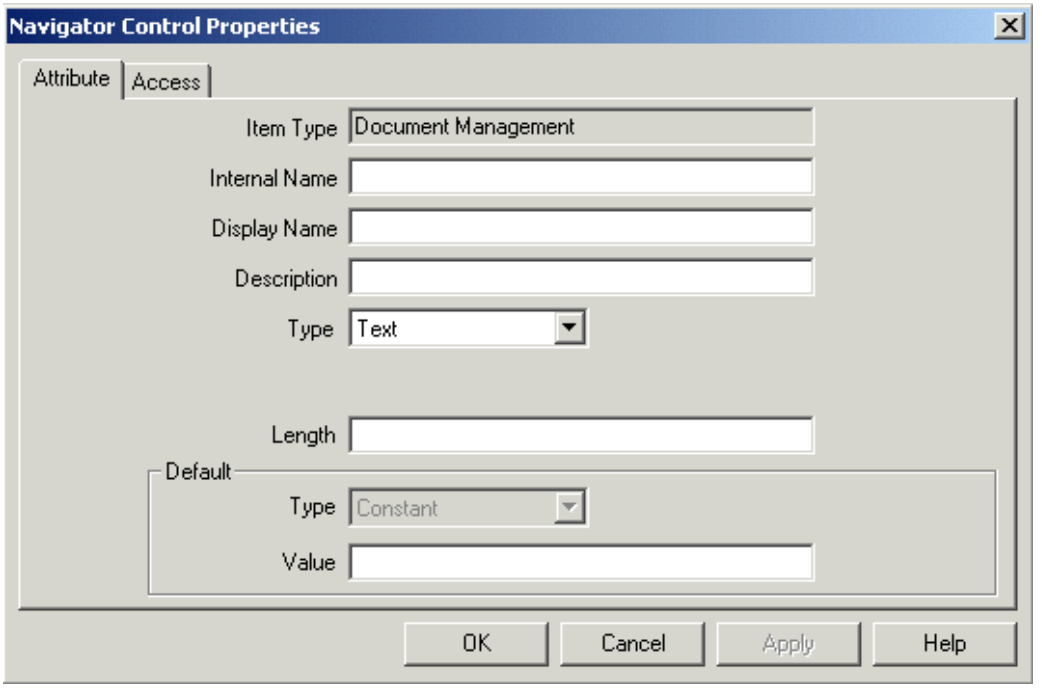
3. Enter a translatable Display Name that is longer and more descriptive. You can also supply a description for the item type.
4. Specify a persistence type of Temporary or Permanent. If you set the persistence type to Temporary, then specify the number of days from the time the item instance completes before its status audit trail can be purged. See: Persistence Type, page 3-4.

5. If your item type has or will have more than one workflow process associated with it, you may specify a selector function using the syntax `<package_name>.<procedure_name>`. The selector function is a PL/SQL stored procedure that automatically identifies the specific process definition the Workflow Engine should execute when a workflow is initiated for this item type. You can also extend the selector function to be a general callback function that resets context information each time the Workflow Engine establishes a new database session to execute activities. See: Standard API for an Item Type Selector or Callback Function, page 6-10.
6. Choose Apply to save your changes.
7. Select the Roles tab page to specify the roles that have access to this item type. (This functionality will be supported in a future release.)
8. Select the Access tab page to set the access and customization levels for this item type. See: Allowing Access to an Object, page 3-17.
9. Choose Apply to save your changes, OK to save your changes and close the property page or Cancel to cancel your changes and close the property page.
10. A secondary branch appears in the navigator tree that represents the item type you just created. You can review or edit the properties of this item type at any time by double-clicking on the item type in the navigator tree or by selecting the item type and choosing Properties from the Edit menu.
11. Define as many item type attributes as necessary to use as global variables in your process. You use these item type attributes to pass values to and from your function, notification, and event activities. See: To Define an Item Type or Activity Attribute, page 3-7.

To Define an Item Type or Activity Attribute:

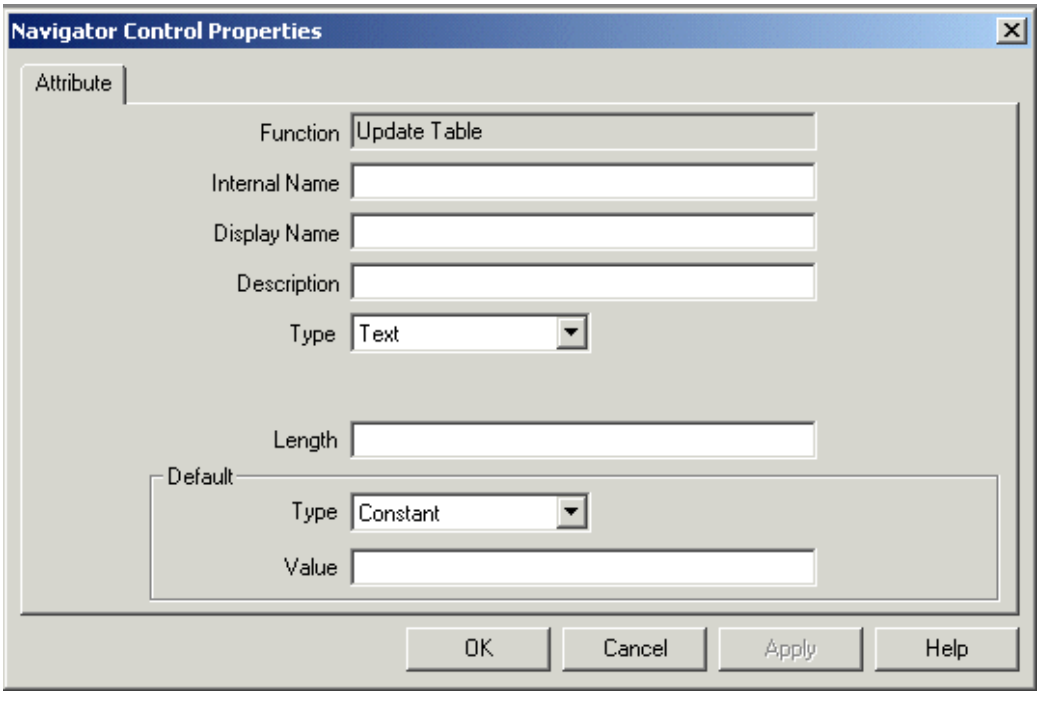
1. To create an item type attribute, select an item type in the navigator tree, then choose New Attribute from the Edit menu.

Item Type Attribute Property Page



To create an activity attribute, select an activity in the navigator tree and choose New Attribute from the Edit menu.

Activity Attribute Property Page



An Attribute property page appears in both cases.

2. Provide an Internal Name in all uppercase with no leading/trailing spaces. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying an attribute.

   Important: To update the internal name for an attribute once it is defined, you must use special SQL scripts called `wfchita.sql` and `wfchacta.sql`. You should only use these scripts to correct errors in an attribute's internal name during design time. Do not use these scripts to rename attributes that are involved in running instances of processes. See: Wfchita.sql, *Oracle Workflow Administrator's Guide* and Wfchacta.sql, *Oracle Workflow Administrator's Guide*.

   Caution: Do not include colons "`:`" or leading/trailing spaces in your internal name.

3. Enter a Display Name. This is the name that appears in the navigator tree.
4. Enter an optional description.
5. Select the data type of the attribute. Form, URL, and document data types are not relevant if you are defining an activity attribute.
6. Depending on the data type of your attribute, provide the following default value information:
   - **Text** - Specify the maximum length of the text attribute and an optional default text string.
   - **Number** - Optionally provide a format mask for your number and a default value.
   - **Date** - Optionally supply a format mask for the date and a default value.
   - **Lookup** - Choose a predefined Lookup Type from which to draw values. Choose a lookup code from that lookup type for the default value.
   - **URL** - Specify an optional Universal Resource Locator (URL) to a network location in the Default Value field and specify the frame target for the URL. See: To Define a URL Attribute, page 3-10.

     Note: The Frame Target field is applicable only for message attributes of type URL. It is not used for item type attributes or activity attributes.

   - **Form** - This attribute is relevant only for the version of Oracle Workflow embedded in Oracle Applications.

     Specify an optional developer form function name and optional argument strings (form function parameters) in the Default Value field. See: Overview of Menus and Function Security, *Oracle Applications Developer's Guide* and To Define a Form Attribute, page 3-12.

   - **Document** - Enter an optional string that identifies the document in the default value field. See: To Define a Document Attribute, page 3-13.

     Note: The Frame Target field is not applicable for attributes of type document. For document attributes, this field is reserved for future use.

- **Role** - Specify a role name. See: Roles, page 4-19.
- **Attribute** - Specify the name of an item type attribute that you want to maintain references to in a process by choosing from the list of existing item type attributes.
- **Event** - If you are defining an item type attribute, you cannot specify any default value for an event attribute. If you are defining an activity attribute, you can only specify an event item type attribute as the default value.

7. For item type attributes, the optional default value is a constant that you enter or select from a list of values. The constant, however, may be a text string that allows for token substitution at runtime.

   For activity attributes, the optional default value may be a constant or an item type attribute. If you want the default to acquire its entire value from an item type attribute, choose Item Attribute in the Default Value region, then use the adjacent poplist field to choose the item type attribute. The item type attribute you select must be associated with the same item type that the activity itself is associated with. The item type attribute you select must also be of the same data type as the activity attribute.

   > **Note:** An activity attribute type of 'Text' is compatible with any item attribute type except the event type. All other activity attribute types must match the item attribute type exactly.

   > **Note:** For attributes of type Lookup, the default value must be a lookup code belonging to that lookup type.

8. Choose Apply to save your changes, OK to save your changes and close the property page, or Cancel to cancel your changes and close the property page.
9. If you are defining an item type attribute, select the Access tab page to set the access levels allowed to modify this attribute. Activity attributes assume the access/protection level of their parent activity. See: Allowing Access to an Object, page 3-17.
10. Choose Apply to save your changes.
11. Any item type attribute you create appears beneath the Attributes branch in the navigator tree. Any function activity attribute you define appears beneath the activity you defined it for in the navigator tree. You can review or edit the properties of an attribute at any time by double-clicking on the attribute in the navigator tree or by selecting the attribute and choosing Properties from the Edit menu.

    > **Important:** The order in which you list these attributes in the navigator tree correlate to the order in which they appear in any list of values that draw upon these attributes. You can use the drag and drop feature of the navigator tree to reorder a set of attributes, or select an attribute and choose Move Attribute Up or Move Attribute Down from the Edit menu.

To Define a URL Attribute:

1. Specify a Universal Resource Locator (URL) to a network location in the Default Value field of the Attribute property page. The URL can be a constant or a value returned from another item attribute.

> **Note:** If you define a URL attribute that points to an Oracle Applications Framework-based Web page, that page must use the self-secured security mode. A page that relies on the user session in its security mode may not be accessible when a user navigates to the URL from an e-mail notification. For more information about Oracle Applications Framework pages, refer to Oracle*MetaLink* note 269138.1, *Oracle Applications Framework Developer's Guide*.

2. You can include argument strings in your URL that are text strings. Additionally, if you are defining a message attribute of type URL, you can include argument strings that are token substituted with other message attributes. The message attributes used for token substitution can have constant values or can reference the values returned from item type attributes. See: To Define a Message Attribute, page 3-46 and To Token Substitute an Attribute, page 3-51.

   To token substitute other message attributes in an argument string, specify the message attributes as follows:

   `-&message_attr-`

   For example, the following string represents a URL with two arguments called `arg1` and `arg2` that are token substituted with the runtime value of message attributes *`msgattr1`* and *`msgattr2`*, respectively:

   `http://www.oracle.com?arg1=-&msgattr1-&arg2=-&msgattr2-`

   > **Note:** If you are defining a message attribute of type URL, you can also include a special token in your argument string called `-&#NID-` which Oracle Workflow substitutes with the notification ID of the runtime notification.

3. If your URL attribute contains an argument string, you must adhere to the following restrictions:
   - You cannot token substitute that argument string with another item attribute of type Document.
   - You can token substitute that argument string with another Form attribute or URL attribute. However, the argument string for the other attribute is not further token substituted.
4. If you need to pass a date and time as an argument to a URL, you should use `TO_CHAR` to format the string as `YYYY/MM/DD+HH24:MI:SS`. Similarly, you need to do the correlating format translation in the function that the URL calls, using `TO_DATE`. This formatting is required because in multibyte databases, the month portion of the `DD-MON-YYYY` format could potentially translate to a value that is not acceptable across a URL.
5. You can also use a URL attribute to include an image in a notification. Define a 'Send' message attribute of type URL that points to an image file with an extension of `gif`, `jpg`, `png`, `tif`, `bmp`, or `jpeg`. Then use a message attribute token to include the URL attribute in the HTML message body. When the notification is viewed from the Notification Details Web page or as an HTML-formatted e-mail, the image appears inline in the message body.

   When defining a URL attribute to point to such an image, you must provide the complete URL for the image, either as a constant or as a value returned from another

item attribute. You cannot token substitute any argument strings within the image URL.

> **Note:** Oracle Workflow displays images only in the message body of a notification. If you check Attach Content for a URL attribute that points to an image file, the URL attribute appears as an attached link, just as other URLs do.
>
> Also, Oracle Workflow displays images only in the HTML-formatted versions of a notification. A plain text e-mail notification displays all URL attributes as the display name of the attribute, followed by a colon and the URL.

6. In Oracle Applications, if you are defining a URL attribute that links to an Oracle Applications Framework page, the page must have the security mode `Self secured`. This security mode is required to ensure that the page can be accessed properly if the URL link is included in an e-mail notification.

   By default, when a user chooses an attached URL link to an Oracle Applications Framework page in a notification, Oracle Applications displays a list of responsibilities assigned to that user that include that page. The user can then select the responsibility with which he or she wants to navigate to the page. When navigating from an e-mail notification, the user must also log into Oracle Applications before selecting a responsibility.

   When you define a URL attribute for an Oracle Applications Framework page, you can optionally specify the responsibility through which you want users to access that page. In this case, users do not need to select a responsibility when they choose the attached URL link; instead, they can navigate directly to the page. However, you must ensure that the users who receive the notification have the specified responsibility assigned to them. Otherwise they will not be able to access the page from that attached URL link.

   To specify a responsibility in the URL attribute, append two special arguments in the argument string for the URL.

   - `#RESP_KEY` - The responsibility key
   - `#APP_SHORT_NAME` - The application short name for the responsibility

   The value for each of these arguments can be a text string enclosed in quotes (`" "`) or can be token substituted with another item type attribute using the following format: `"&ITEM_ATTR"`

7. Choose OK when you are done.

To Define a Form Attribute:

1. Specify a developer form function name and any optional argument string (form function parameters) in the Default Value field of the form Attribute property page.
2. The default value must be entered using the following format:

   ```
   function_name:arg1=value1 arg2=value2 ...argN=valueN
   ```

   The value of *argN* can be a text string enclosed in quotes (`" "`) or can be token substituted with another item type attribute in any of the following ways, where *&item_attr* represents the internal name of the item type attribute:

- *argN="&item_attr"*
- *argN="Value &item_attr"*

See: To Token Substitute an Attribute, page 3-51.

> **Note:** If you are defining a message attribute of type Form, you can also include a special token in your argument string called `&#NID` which Oracle Workflow substitutes with the notification ID of the runtime notification.

3. If your form attribute contains an argument string, you must adhere to the following restrictions:
   - You cannot token substitute the value of *argN* with another item attribute of type Document.
   - You can token substitute the value of that argument string with another Form attribute or URL attribute; however, the argument string for the other attribute is not further token substituted.
4. By default, when a user chooses an attached form link in a notification, Oracle Applications displays a list of responsibilities assigned to that user that include that form. The user can then select the responsibility with which he or she wants to navigate to the form. When navigating from an e-mail notification, the user must also log into Oracle Applications before selecting a responsibility.

   When you define a form attribute, you can optionally specify the responsibility through which you want users to access that form. In this case, users do not need to select a responsibility when they choose the attached form link; instead, they can navigate directly to the form. However, you must ensure that the users who receive the notification have the specified responsibility assigned to them. Otherwise they will not be able to access the form from that attached form link.

   To specify a responsibility in the form attribute, append two special arguments in the argument string for the form function.

   - `#RESP_KEY` - The responsibility key
   - `#APP_SHORT_NAME` - The application short name for the responsibility

   The value for each of these arguments can be a text string enclosed in quotes (`" "`) or can be token substituted with another item type attribute using the following format: `"&ITEM_ATTR"`

   For example, the following form attribute value specifies that the Users form should be opened from the System Administration responsibility.

   ```
   FND_FNDSCAUS:#RESP_KEY="SYSTEM_ADMINISTRATION" #APP_SHORT_NAME="S
   YSADMIN"
   ```

5. Choose OK when you are done.

To Define a Document Attribute:

1. Enter a string that identifies the document in the default value field of the Attribute property page.

   You can identify the following types of document for a document attribute:

- A PL/SQL document
- A PL/SQL CLOB document
- A PL/SQL BLOB document
- An Oracle Applications Framework region (Oracle Applications only)

2. A PL/SQL document represents data from the database as a character string, generated from a PL/SQL procedure. Specify the default value of a PL/SQL document as

   `plsql:<procedure>/<document_identifier>`

   Replace `<procedure>` with the PL/SQL package and procedure name, separated by a period. Replace `<document_identifier>` with the PL/SQL argument string that you want to pass directly to the procedure. The argument string should identify the document.

   > **Note:** The PL/SQL procedure must follow a standard API format. See: Standard APIs for "PL/SQL" Documents, page 6-13.

   For example, the following string represents the PL/SQL document po_req:2034, generated by the procedure po_wf.show_req.

   ```
   plsql:po_wf.show_req/po_req:2034
   ```

   > **Note:** The maximum length of the data that a PL/SQL document can contain is 32 kilobytes. If you expect your document to exceed 32 Kb, you should use a PL/SQL CLOB document to hold the data instead.

   A PL/SQL document can either be displayed in the message body of a notification or included as an attachment to the notification.

3. A PL/SQL CLOB document represents data from the database as a character large object (CLOB), generated from a PL/SQL procedure. Specify the default value of a PL/SQL CLOB document as

   `plsqlclob:<procedure>/<document_identifier>`

   Replace `<procedure>` with the PL/SQL package and procedure name, separated by a period. Replace `<document_identifier>` with the PL/SQL argument string that you want to pass directly to the procedure. The argument string should identify the document.

   > **Note:** The PL/SQL procedure must follow a standard API format. See: Standard APIs for "PL/SQL" Documents, page 6-13.

   For example, the following string represents the PL/SQL CLOB document, po_req:2036, generated by the procedure po_wf.show_req_clob.

   ```
   plsqlclob:po_wf.show_req_clob/po_req:2036
   ```

   A PL/SQL CLOB document that contains text or HTML can either be displayed in the message body of a notification or included as an attachment to the notification.

   PL/SQL CLOB documents do not support further substitution of message attribute tokens. The text or HTML contents of the CLOB are printed in a message body just as they are generated by the PL/SQL procedure.

- Do not use tokens within the CLOB.
- Ensure that the PL/SQL procedure performs any formatting you require.

A PL/SQL CLOB document that you include as an attachment to a notification can also contain a PDF or RTF document or, if your database version is Oracle9*i* Database or higher, other binary data encoded to base64. You should first store the document in the database as a binary large object (BLOB) and then convert the document into a CLOB as part of the PL/SQL procedure you use to produce the CLOB. See: Standard APIs for "PL/SQL" Documents, page 6-13.

You can use the UTL_RAW.Cast_To_VARCHAR2 function to convert PDF or RTF data from the BLOB into VARCHAR2 data that you write to a CLOB. If your database version is Oracle9*i* Database or higher, you can optionally use the WF_MAIL_UTIL.EncodeBLOB procedure to encode the binary data to base64. See: UTL_RAW, *Oracle Supplied PL/SQL Packages and Types Reference* and EncodeBLOB, *Oracle Workflow API Reference*.

> **Note:** Oracle8*i* Database does not support base64 encoding, so if you are using Oracle8*i* Database, the WF_MAIL_UTIL.EncodeBLOB procedure is not available, and you cannot store binary data other than PDF or RTF documents in a PL/SQL CLOB document. This feature is available only if you are using Oracle9*i* Database and higher.
>
> However, both Oracle8*i* Database and Oracle9*i* Database and higher support the UTL_RAW.Cast_To_VARCHAR2 function, so you can store PDF and RTF documents in an attached PL/SQL CLOB document on any of these database versions.

4. A PL/SQL BLOB document represents data from the database as a binary large object (BLOB), generated from a PL/SQL procedure. Specify the default value of a PL/SQL BLOB document as

   `plsqlblob:<procedure>/<document_identifier>`

   Replace `<procedure>` with the PL/SQL package and procedure name, separated by a period. Replace `<document_identifier>` with the PL/SQL argument string that you want to pass directly to the procedure. The argument string should identify the document.

   > **Note:** The PL/SQL procedure must follow a standard API format. See: Standard APIs for "PL/SQL" Documents, page 6-13.

   For example, the following string represents the PL/SQL BLOB document po_req:2038, generated by the procedure po_wf.show_req_blob.

   ```
   plsqlblob:po_wf.show_req_blob/po_req:2038
   ```

   PL/SQL BLOB documents do not support further substitution of message attribute tokens.

   A PL/SQL BLOB document can contain an image or an application file stored as binary data, such as a PDF or RTF document or other types of files. You can include a PL/SQL BLOB document as an attachment to a notfication. However, this type of document cannot be displayed in the message body of a notification.

5. If you wish to generate the document identifier for a PL/SQL, PL/SQL CLOB, or PL/SQL BLOB document dynamically, you can token substitute the document identifier with other item type attributes. The item attribute names must be in uppercase and must be separated by a colon. See: To Token Substitute an Attribute, page 3-51.

   For example:

   `plsql:po_wf.show_req/&ITEM_ATTR1:&ITEM_ATTR2`

   > **Note:** If you are defining a message attribute of type Document, you can also include a special token in your argument string called `&#NID` which Oracle Workflow substitutes with the notification ID of the runtime notification.

6. You can also use an attribute of type document to reference an Oracle Applications Framework region that you want to embed in a notification message. Specify the JSP call to reference the region as the default value for the attribute. See: Embedding Oracle Applications Framework Regions in Messages, page 3-38.

   An Oracle Applications Framework region can only be displayed within the message body of a notification. It cannot be included as an attachment to the notification.

7. Choose OK when you are done.

To Copy an Item Type:

1. Select the item type to copy in the navigator tree.
2. Drag the item type, holding down your select mouse button, to the data store or workspace you want to copy it to.

   You can also use the Copy and Paste commands in the Edit menu.

3. If you copy this item type back to the same data store, you get prompted to enter a new internal and display name for the item type in the Item Type property page. This is because every item type must have a unique internal and display name. When you are done, choose OK.

   Note that when you copy an item type, you also copy all the components associated with the item type. Since most components must also have unique internal and display names, you may get prompted to update those components' internal and display names in their property pages as well.

4. If you copy an item type to a data store where a previous version of the same item type already exists, you update the existing version of the item type in that target data store with the changes in the version of the item type you are copying.

   > **Important:** The order in which you drag two or more item types to a new store is important. For example, suppose an item type references objects in the Standard item type. If you plan to copy that item type and the Standard item type to a new data store, you should first drag the Standard item type to the new data store before dragging the other item type over; otherwise the other item type will have unresolved references to the Standard item type.

To Copy an Attribute:

1. Select the attribute to copy in the navigator tree.
2. Drag the attribute, holding down your select mouse button, to the component branch you want to copy it to.
3. If you copy an attribute to a component associated with the same item type, the property page for the attribute appears.

   Enter a new unique internal name and display name for the attribute.

   When you are done, choose OK.

   Note: You can also use the Copy and Paste options in the Edit menu.

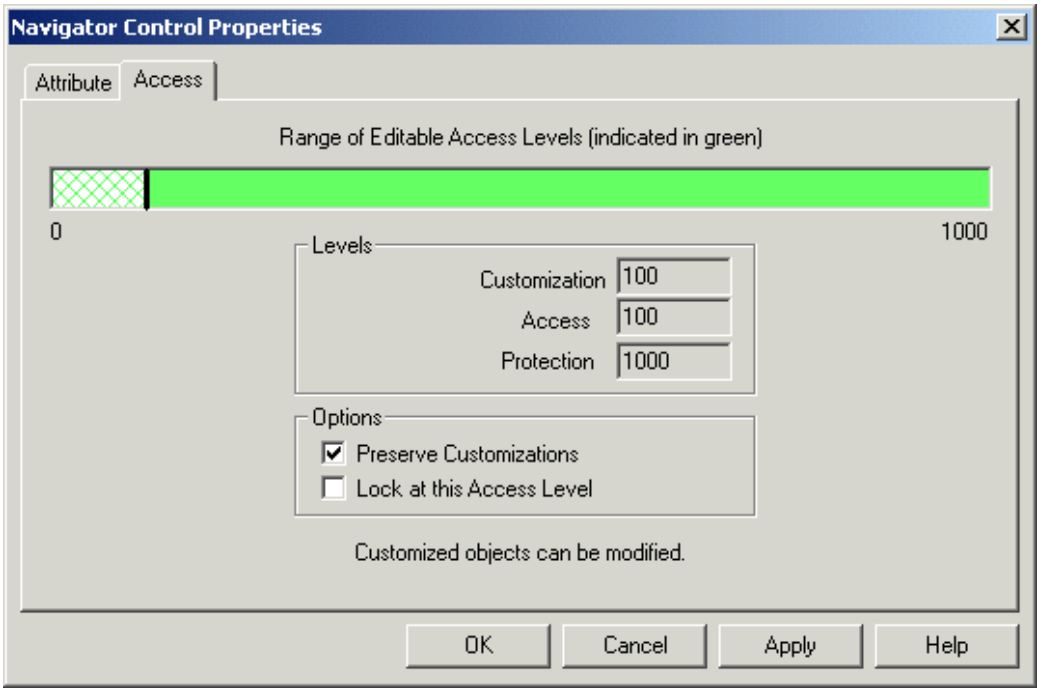
Related Topics

Using the Edit Button in a Property Page, page 2-6

To Define a Message Attribute, page 3-46

Allowing Access to an Object

Access Property Page



In the Access tab page, the 'Range of Editable Access Levels' indicator bar provides a relative indication of the range of access levels that can edit the object. The shaded area represents the access levels that can edit the object, while the vertical bar represents your current access level. See: Overview of Oracle Workflow Access Protection, *Oracle Workflow Administrator's Guide*.

The indicator bar can be shaded solid green, or shaded with any combination of solid green and crosshatch grey. If the "Allow modifications of customized objects" check box in the "About Oracle Workflow Builder" dialog box of the Help menu is:

- Checked - The range of editable access levels can appear as a combination of solid green and crosshatch grey areas. The levels depicted by grey crosshatches represent levels that usually cannot modify customized objects, but can now do so because Oracle Workflow Builder is operating in 'upload' mode. Upload mode means that Oracle Workflow Builder can save your edits, overwriting any protected objects that you have access to modify as well as any previously customized objects.
- Unchecked - The range of editable access levels appears as a solid green area. This indicates that when you save your work, Oracle Workflow Builder is operating in 'upgrade' mode, only saving edits to protected objects that you have access to change and leaving objects that have been previously customized untouched.

These two modes match the upgrade and upload behavior of the Workflow Definitions Loader program. See: To Set the Access Level for an Object, page 3-18 and Using the Workflow Definitions Loader, *Oracle Workflow Administrator's Guide*.

To Set the Access Level for an Object:

1. Select the Access tab of the property page.
2. In the Options region, use the 'Preserve Customizations' and 'Lock at this Access Level' check boxes to define the access levels that can modify this object. The options that you check in this region directly affect the values that appear in the Levels region.

   The following table illustrates how the customization and protection levels of an object are affected when you check different combinations of these options. This table assumes that the user setting the options has an access level of 100.

Results of Access Option Combinations

Selected Checkbox	Resulting Levels	Levels Allowed to Modify the Object
NONE - Object can be updated at any time, by anyone.	Customization = 0, Access = 100, Protection = 1000	Object may be updated by any access level (0-1000).
Preserve Customizations - Disallow customized objects from being overwritten during a workflow definition upgrade.	Customization = 100, Access = 100, Protection = 1000	Object may only be updated by access levels from 100-1000. If the "Allow modifications of customized objects" checkbox is checked, customized objects can also be updated by access levels 0-99, as represented by grey crosshatches in the indicator bar.
Lock at this Access Level - Protect the object at the current access level and do not allow the object to be customized.	Customization = 0, Access = 100, Protection = 100	Object may only be updated by access levels from 0-100.
BOTH - Object can only be updated by the access level at which the object is protected.	Customization = 100, Access = 100, Protection = 100	Object cannot be updated by any access level other than 100. If the "Allow modifications of customized objects" checkbox is checked, customized objects can also be updated by access levels 0-99, as represented by grey crosshatches in the indicator bar.

3. Choose the Apply button to save your changes.

   Note: An object appears with a small red lock over its icon in the navigator tree to indicate that it is a read-only object if you are operating at an access level that does not have permission to edit an object, that is, your access level is in a white area of the 'Range of Editable Access Levels' indicator bar.

Lookup Types

A lookup type is a static list of values. These lists can be referenced by activities and by item type, message or activity attributes. For example, an activity can reference a lookup type for its possible result values, while a message attribute can reference a lookup type as a means of providing a list of possible responses to the performer of a notification.

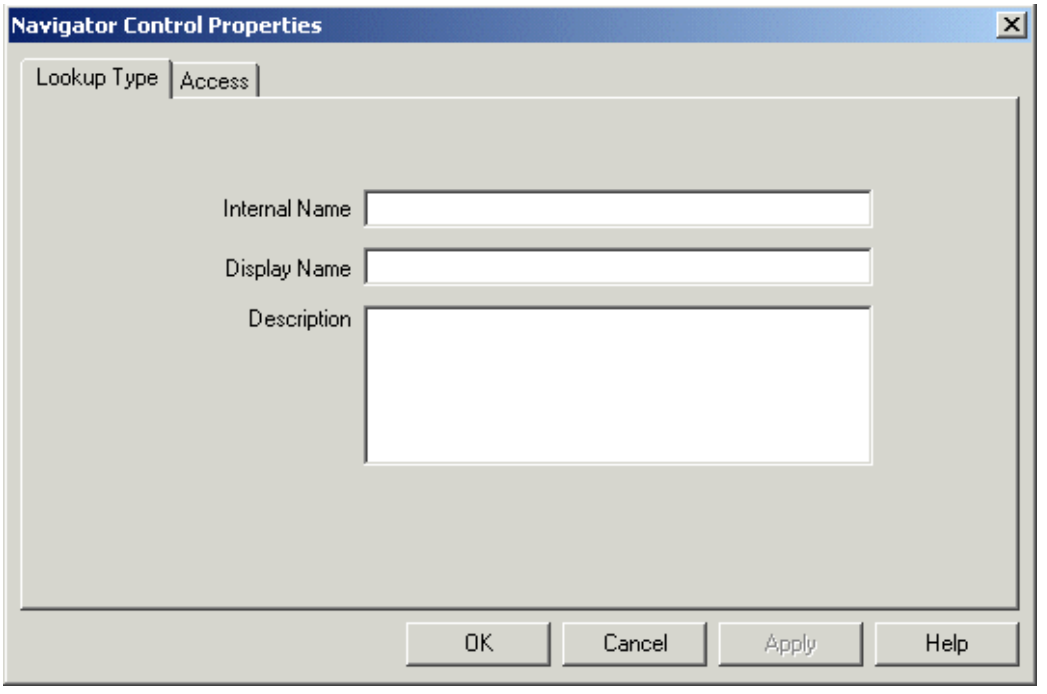
When you define a lookup type, you associate it with an particular item type. However, lookup types are not item type specific; when you create an activity or an attribute, you can reference any lookup type in your current data store, regardless of the item type that the lookup type is associated with.

Defining Lookup Types

To Create Lookup Types:

1. Select an item type from the navigator tree and choose New Lookup Type from the Edit menu. A Lookup Type property page appears.

Lookup Type Property Page



2. Lookup types have an all-uppercase Internal Name with no leading/trailing spaces and a translatable Display Name. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying a lookup type.

   Important: To update the internal name for a lookup type once it is defined, you must use a special SQL script called `wfchlut.sql`. You should only use this script to correct errors in a lookup type's internal name during design time. Do not use this script to rename lookup types that are involved in running instances of processes. See: Wfchlut.sql, *Oracle Workflow Administrator's Guide*.

   Caution: Do not include colons "`:`" or leading/trailing spaces in your internal name.

   You can supply an optional description for this lookup type.

3. Select the Access tab page to set the access levels allowed to modify this lookup type. See: Allowing Access to an Object, page 3-17.

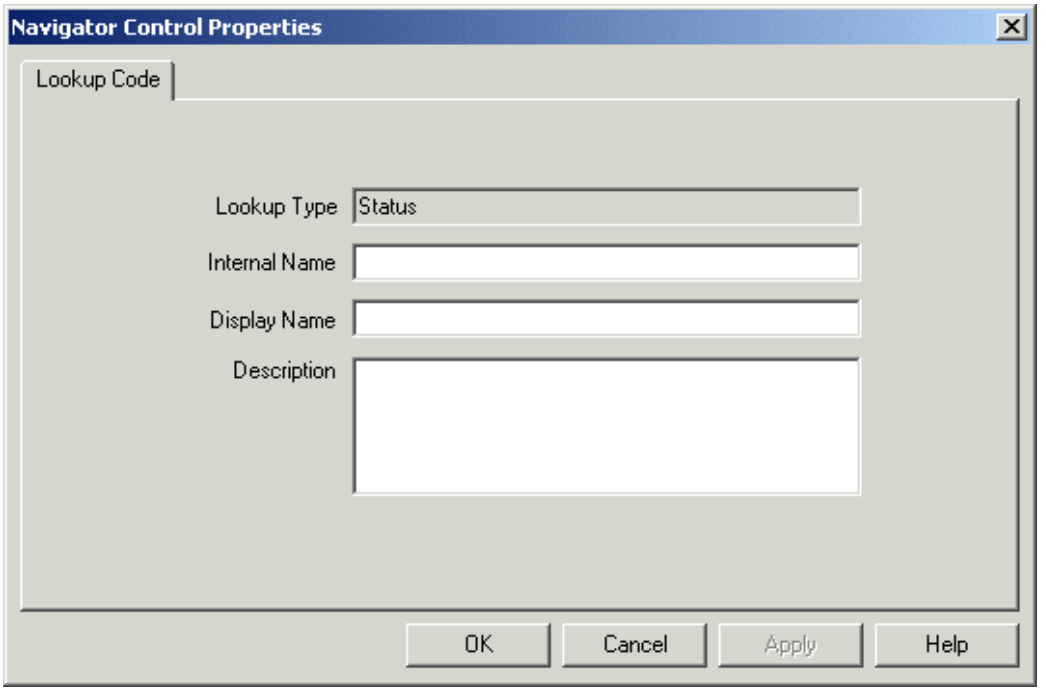
4. Choose Apply to save your changes, OK to save your changes and close the property page, or Cancel to cancel your changes and close the property page.

5. The lookup type you just defined now appears beneath the Lookup Types branch in the navigator tree. You can review or edit the properties of this lookup type at any time by double-clicking on the lookup type in the navigator tree or by selecting the lookup type and choosing Properties from the Edit menu.
6. Now define the lookup codes for your lookup type. See: To Create Lookup Codes for a Lookup Type, page 3-21.

To Create Lookup Codes for a Lookup Type:

1. Select a lookup type from the navigator tree and choose New Lookup Code from the Edit menu. A Lookup Code property page appears.

Lookup Code Property Page



2. Enter an Internal Name with no leading/trailing spaces and a Display Name for the lookup code. You can also enter an optional description. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying a lookup code.

> **Important:** To update the internal name for a lookup code once it is defined, you must use a special SQL script called `wfchluc.sql`. You should only use this script to correct errors in a lookup code's internal name during design time. Do not use this script to rename lookup codes that are involved in running instances of processes. See: Wfchluc.sql, *Oracle Workflow Administrator's Guide*.

> **Caution:** Do not include colons "`:`" or leading/trailing spaces in your internal name.

3. Choose Apply to save your changes, OK to save your changes and close the property page, or Cancel to cancel your changes and close the property page.
4. The lookup code you just defined now appears beneath the lookup type you created it for in the navigator tree. You can review or edit the properties of this lookup code at any time by double-clicking on the lookup code in the navigator tree or by selecting the lookup code and choosing Properties from the Edit menu.
5. Repeat steps 1, 2, and 3 if you wish to create additional lookup codes for a specific lookup type.

To Copy a Lookup Type:

1. Select the lookup type to copy in the navigator tree.
2. Use the Copy and Paste options in the Edit menu to copy and paste the lookup type to the item type you want. You can also drag and drop the lookup type to the item type you want.
   - If you copy a lookup type back to the same item type, or if you copy a lookup type to an item type in a different data store when the lookup type already exists in any item type in that data store, then the property page appears for you to enter a unique internal and display name for the new lookup type. When you are done, choose OK.
   - If you copy a lookup type to an item type in a different data store, and that lookup type does not yet exist in any item type in that data store, then the new lookup type is copied with the same internal and display name as the original. You do not have to enter new names.

     Note: Copying a lookup type also copies any lookup codes assigned to it.

     Note: You cannot use the Copy and Paste options to copy a lookup type to another item type in the same data store. However, you can drag and drop a lookup type to another item type in the same data store; by doing so, you actually move the lookup type to the new item type.

To Copy a Lookup Code:

1. Select the lookup code to copy in the navigator tree.
2. Hold down your mouse select button as you drag the lookup code to the lookup type you want to copy it to.
3. If you drag the lookup code to the same lookup type or to a lookup type where this code already exists, then when you release your mouse button, a properties page appears for you to enter a unique internal and display name the new lookup code. When you are done, choose OK.

   Note: You can also use the Copy and Paste options in the Edit menu.

Messages

The Messages branch of the navigator tree lists all available workflow messages for the current item type.

A message is what a notification activity sends to a role in a workflow process. A message can prompt a user for a reply or an action to take that determines what the next activity in the process should be. The recipient of a workflow message is called the performer.

Each message is associated with a particular item type. This allows the message to reference the item type's attributes for token replacement at runtime when the message is delivered.

When you define a message, you can specify that the message prompts a recipient for a special response value that the Workflow Engine then uses to determine how to branch to the next eligible activity in the process. You can create a message with context-sensitive content by including message attribute tokens in the message subject and body that reference item type attributes. You can also use message attribute tokens to embed inline images in the message body. A message function lets you include a formatted table of message attributes or an action history table in the message body. You can also attach message attributes that represent entire documents or URLs to a notification message. In addition, you can create message attributes that generate a response section that is unique to the message. In Oracle Applications, you can also embed an Oracle Applications Framework region within the message body.

You can drag a message onto the Notifications branch to create a new notification activity that sends that message. You can also drag a message directly onto an existing notification activity to update the message that the activity sends.

Message Result

When you create a message for a notification activity, you should make note of whether the notification activity has a Result Type specified. If it does, then the message you create needs to prompt the notification recipient for a special response that is interpreted as the result of the notification activity. The Workflow Engine uses that result to determine how it should branch to the next eligible activity in the process.

To create a message that prompts for this special response, complete the Result tab in the message's property page. The information you enter creates a special 'Respond' message attribute for the message that has an internal name of `RESULT`. The `RESULT` message attribute has a data type of Lookup and must be set to the same lookup type as the notification activity's Result Type. This ensures that the performer of the notification can choose from a list of possible response values that matches the list of possible results that the notification activity is expecting. See: Send and Respond Message Attributes, page 3-23.

Send and Respond Message Attributes

Once you create a message, you can define as many message attributes as necessary for that message. Message attributes are listed beneath a message in the navigator tree.

The source (Send or Respond) of a message attribute determines how the message attribute is used. When you define a message attribute with a source of 'Send', you can embed the message attribute in the subject and/or body of the message for token substitution. In addition, you can attach the message attribute to the message when the notification is sent.

Each message attribute has a specific data type. The value of a 'Send' message attribute can be a constant or can be a value returned by an item type attribute of that same data type. To embed a message attribute in a message's subject or body for token

substitution, specify the internal name of the message attribute using the format `&MESGATTR` within the subject or body text.

> **Note:** You should not embed a message attribute of type Document in a message's subject, since Document message attributes cannot be token substituted in the subject. Document message attributes embedded in the subject will be ignored.
>
> You can, however, embed Document message attributes within the body of a message for token substitution.

A message attribute defined with a source of 'Respond' constitutes the response section of a message. A 'Respond' message attribute provides instructions that prompts a recipient for a response. When you define a 'Respond' message attribute, you must specify the data type of the attribute. You can also provide an optional default value for the response. The default value can be a constant or a value returned from an item type attribute of the same data type.

You can define 'Respond' message attributes of type URL or form if you want the notification recipient to respond to the notification through the specified Web page or form. In this case that custom Web page or form must mark the notification as closed and include a call to the Workflow Engine *CompleteActivity()* API to inform the Workflow Engine when the notification response is complete. Also, you must not define any 'Respond' message attributes of other types. Any response values that are required must be entered in the specified Web page or form. See: CompleteActivity, *Oracle Workflow API Reference*.

In Oracle Applications, if you define 'Respond' message attributes of type URL or form, the Notification Details page will display only the URL response links or form response icons and will not display any other response fields. However, users will still be able to reassign or request more information for the notification as usual, if the notification is defined to allow those options.

See: Message Attributes, *Oracle Workflow Administrator's Guide*.

Action History

A notification can include an action history table that shows what actions users have performed on the notification to date. The action history table contains a row for each previous execution of the same notification activity node in a process, as well as a row for the initial submission of the process. For example, for a requisition approval notification activity that sends a certain notification to several approvers in turn, the action history table would contain a row for each approver to whom the notification was sent, as well as a row for the process owner. Additionally, if a notification is reassigned to another user in either delegate or transfer mode, or if one user requests more information about the notification and another user answers that request, the action history contains rows for those actions also.

- If the notification requires a response, then Oracle Workflow automatically includes the action history table in the notification.
- If the notification does not require a response, then Oracle Workflow only includes the action history table automatically if the notification has been reassigned at least once or if at least one recipient has requested more information about the notification. However, you can manually include the action history table by calling the special message function `WF_NOTIFICATION(HISTORY)` in the message body.

The formatting of the action history table depends on whether the notification contains an embedded Oracle Applications Framework region.

- If the notification contains an Oracle Applications Framework region, or if it contains only plain text and calls to the special `WF_NOTIFICATION()` message function, the table is formatted as an Oracle Applications Framework region.
- If the notification does not contain any Oracle Applications Framework regions, but does contain references to other types of message attributes, then the table is formatted in a standard Oracle Workflow format. This category includes all notifications in standalone Oracle Workflow as well as any Oracle Applications notifications that reference values other than Oracle Applications Framework regions.

The standard action history table provided by Oracle Workflow includes the following columns:

- Num - A sequence number indicating the order in which the actions on this notification took place, beginning at one (1) for the initial submission of the process by the process owner.
- Action Date - The date that a user acted on the notification.
- Action - The action performed on the notification.
- From - The user who performed the action.
- To - The next user who must act on the notification as a result of this action. For example, the next approver, the user to whom the notification was reassigned, or the user from whom more information was requested.
- Details - An additional note from the user who performed the action. To allow a recipient to add a note for the action history table while responding, you must create a special 'Respond' message attribute with the internal name `WF_NOTE`.

  Define the message attribute with the following properties:

  - Internal Name - `WF_NOTE`
  - Display Name - Note
  - Description - Note
  - Type - Text
  - Source - Respond

  When the `WF_NOTE` attribute is defined with a source of 'Respond', it appears as part of the notification response section, and the recipient can enter a note there when responding to the notification. Oracle Workflow retrieves the note text stored in the `WF_NOTE` attribute and displays it in the action history table.

  If the responder did not enter a note, or if the `WF_NOTE` message attribute was not defined for the notification, then the Note column in the action history table is left blank.

  Oracle Workflow also lets users enter notes while reassigning a notification, requesting more information, or answering a request for more information. These notes are likewise displayed in the action history table.

To allow the process owner to add a note for the action of submitting the process and sending the initial notification, you must define a special message attribute named `#SUBMIT_COMMENTS`. See: #SUBMIT_COMMENTS Attribute, page 3-33.

You can optionally create a custom action history table to replace the standard action history table, using the special message attribute named `#HISTORY`. Your custom action history table must be formatted as an Oracle Applications Framework region, PL/SQL document, or PL/SQL CLOB document. See: #HISTORY Attribute, page 3-33, Embedding Oracle Applications Framework Regions in Messages, page 3-38, and WF_NOTIFICATION() Message Function, page 3-26.

WF_NOTIFICATION() Message Function

In addition to message attribute tokens, you can also use a special message function called `WF_NOTIFICATION()` to add context-sensitive content to a message body. Depending on the parameters you provide, the `WF_NOTIFICATION()` function can produce either a table of message attributes or an action history table for the notification.

- In Oracle Applications, the tables are created as Oracle Applications Framework regions, unless the message body also includes any message attribute tokens that reference values other than Oracle Applications Framework regions. In this case, the tables are created in a standard Oracle Workflow format. See: Embedding Oracle Applications Framework Regions in Messages, page 3-38.
- In standalone Oracle Workflow, the tables are always created in the standard Oracle Workflow format.

> **Note:** `WF_NOTIFICATION()` is not a PL/SQL function, but rather a special message function that can only be called within an Oracle Workflow message body.

Message Attribute Table

To include a table of message attributes in a message body, call `WF_NOTIFICATION()` with the `ATTRS` option followed by the internal names of the message attributes, separated by commas. Use the following format:

```
WF_NOTIFICATION(ATTRS,<attribute1>,<attribute2>,<attribute3>,...)
```

> **Note:** You must not include any spaces or carriage returns in the call to `WF_NOTIFICATION()`. You only need to use a comma to delimit the parameters in the list.

The message attribute table contains a row for each message attribute listed in the `WF_NOTIFICATION()` call, showing the display name and the value for each attribute.

Action History Table

To include an action history table in a message body, call `WF_NOTIFICATION()` with the `HISTORY` option in the following format:

```
WF_NOTIFICATION(HISTORY)
```

The action history table contains the standard action history information provided by Oracle Workflow, unless you have defined the `#HISTORY` attribute for a message in

Oracle Applications to use a custom action history region instead. See: Action History, page 3-24 and #HISTORY Attribute, page 3-33.

#WF_REASSIGN_LOV Attribute

In Oracle Applications, you can use a special message attribute with the internal name `#WF_REASSIGN_LOV` to specify the users to whom a message can be reassigned. Create a role whose members are all the users that you want to allow as possible new recipients when the notification is reassigned, and assign this role as the value for the `#WF_REASSIGN_LOV` attribute. Then, when the notification recipient attempts to reassign the notification, only the users that belong to that role will appear in the list of values for the Assignee field. See: To Reassign a Notification to Another User (for Oracle Applications), *Oracle Workflow User's Guide.*

The `#WF_REASSIGN_LOV` attribute must be of type role and must have a source of Send. You can either specify a particular role as a constant value for the `#WF_REASSIGN_LOV` attribute, or specify an item type attribute as the value and include logic in your workflow process that dynamically sets that item type attribute to the role you want at runtime. If no existing role meets your needs, you can include logic in your process to create an ad hoc role at runtime and add the users you want to that role. See: CreateAdHocRole, *Oracle Workflow API Reference* and AddUsersToAdHocRole, *Oracle Workflow API Reference*.

> **Note:** The Worklist pages that support the `#WF_REASSIGN_LOV` attribute are currently available only for the version of Oracle Workflow embedded in Oracle Applications.

#HIDE_REASSIGN Attribute

You can use a special message attribute with the internal name `#HIDE_REASSIGN` to hide the Reassign button in the Notification Details Web page. The response section in the Notification Details page includes the Reassign button by default. If you want to restrict users from reassigning a notification, you can add the `#HIDE_REASSIGN` attribute to control whether the Reassign button is displayed or hidden. See: To Reassign a Notification to Another User (for standalone Oracle Workflow), *Oracle Workflow User's Guide* or To Reassign a Notification to Another User (for Oracle Applications), *Oracle Workflow User's Guide*.

> **Note:** In Oracle Applications, the Notification Details page may include the Delegate button or the Transfer button instead of the Reassign button, depending on the setting of the WF: Notification Reassign Mode profile option. The `#HIDE_REASSIGN` attribute controls the Delegate button and the Transfer button in the same way as the Reassign button. See: Setting the WF: Notification Reassign Mode, *Oracle Workflow Administrator's Guide*.

In Oracle Applications, the `#HIDE_REASSIGN` attribute also controls whether a notification can be reassigned using the Reassign button in the Advanced Worklist, Personal Worklist, or self-service home page. The Reassign button in these pages will still be displayed, but an error message will appear if users attempt to reassign notifications for which the `#HIDE_REASSIGN` attribute has been set. See: To View Notifications from the Advanced Worklist, *Oracle Workflow User's Guide* and To View Notifications from the Personal Worklist, *Oracle Workflow User's Guide*.

Additionally, with this attribute you can specify whether or not the notification can still be reassigned through vacation rules. You can choose to both hide the Reassign button and prevent reassignment through vacation rules. In this case Oracle Workflow does not apply any vacation rules to reassign those notifications, but simpy delivers the notifications to the worklist of the original recipient. You can also choose to hide the Reassign button but still allow reassignment through vacation rules.

The `#HIDE_REASSIGN` attribute must be either of type text or lookup.

- To hide the Reassign button in the Notification Details page, and to prevent reassignment from the Advanced Worklist, Personal Worklist, and self-service home page, as well as through vacation rules, set the value of this attribute to `Y`.
- To hide the Reassign button in the Notification Details page, and to prevent reassignment from the Advanced Worklist, Personal Worklist, and self-service home page, but still allow reassignment through vacation rules, set the value of this attribute to `B`.
- To display the Reassign button in the Notification Details page, and to allow reassignment from the Advanced Worklist, Personal Worklist, self-service home page, and vacation rules, set the value to `N`.

  Note: If you want to prevent reassignment in all cases, it is recommended to define the `#HIDE_REASSIGN` attribute with a type of lookup and assign it the predefined Yes/No lookup type that is provided in the Standard item type. The Yes/No lookup type contains two lookup codes with the display names Yes and No, representing the values `Y` and `N`, respectively. However, if you want to set the value to `B`, you must either define the attribute with a type of text or define a custom lookup.

- If you always want to restrict reassignment for notifications using a particular message, specify the value `Y` or `B` as a constant.
- If you only want to restrict reassignment in certain cases, specify an item type attribute as the value. Then include logic in your workflow process that dynamically determines at runtime whether reassignment should be permitted or restricted and sets the item type attribute to `Y`, `B`, or `N`, accordingly.

#HIDE_MOREINFO Attribute

In Oracle Applications, you can use a special message attribute with the internal name `#HIDE_MOREINFO` to hide the Request Information button in the Notification Details page. When users view a notification that requires a response from their Worklist page, the response region in the Notification Details page includes the Request Information button by default. If you want to prevent users from requesting more information about a notification, you can add the `#HIDE_MOREINFO` attribute to control whether the Request Information button is displayed or hidden.

Note: In both the standalone version of Oracle Workflow and the Oracle Applications version of Oracle Workflow, the `#HIDE_MOREINFO` attribute also determines whether the Request Information response link is included or excluded in HTML-formatted e-mail notifications. However, note that if the `#HIDE_MOREINFO` attribute is not defined for a particular notification, the default behavior is different for the Notification Details page and for HTML-formatted e-mail notifications. The Notification Details page displays the

Request Information button if the #HIDE_MOREINFO attribute is not defined, while an HTML-formatted e-mail notification will exclude the Request Information response link by default if the #HIDE_MOREINFO attribute is not defined. To control this option in all interfaces where users access notifications, you should explicitly define and set the #HIDE_MOREINFO attribute. See: To Respond to an HTML E-mail Notification, *Oracle Workflow User's Guide*

The #HIDE_MOREINFO attribute must be either of type text or lookup. To hide the Request Information button, set the value of this attribute to Y. To display the Request Information button, set the value to N.

Note: It is recommended to define the #HIDE_MOREINFO attribute with a type of lookup and assign it the predefined Yes/No lookup type that is provided in the Standard item type. The Yes/No lookup type contains two lookup codes with the display names Yes and No, representing the values Y and N, respectively.

- If you always want to hide the Request Information button for notifications using a particular message, specify the value Y as a constant.
- If you only want to hide the Request Information button in certain cases, specify an item type attribute as the value. Then include logic in your workflow process that dynamically determines at runtime whether the button should be hidden or displayed and sets the item type attribute to Y or N, respectively.

Note: The Worklist pages that include the Request Information button are currently available only for the version of Oracle Workflow embedded in Oracle Applications.

#FROM_ROLE Attribute

You can use a special message attribute with the internal name #FROM_ROLE to specify the role that is the source of a notification. For example, if you have a notification that informs an approver that a requisition was submitted, you can set the requisition preparer as the From Role for the message.

If a notification with this attribute is reassigned through the Worklist Web pages, Oracle Workflow automatically sets the From Role to the previous recipient when sending the notification to the new recipient.

The From Role for each notification is displayed in the Worklist Web page and in e-mail notifications to give users additional information for reviewing and responding to the notifications. Additionally, the Find Notifications page, administrator Notification Search page, and Personal Worklist let you search for notifications based on the From Role. See: To View Notifications from the Worklist, *Oracle Workflow User's Guide*, and To Find Notifications, *Oracle Workflow User's Guide*.

The #FROM_ROLE attribute must be of type role and must have a source of Send. The display name and description of the attribute should both be From Role. You should specify an item type attribute as the value for the #FROM_ROLE attribute and include logic in your workflow process that dynamically sets that item type attribute to the role you want at runtime.

#WF_SIG_POLICY Attribute

In Oracle Applications, you can use a special message attribute with the internal name `#WF_SIG_POLICY` to require that a user's response to a notification be signed electronically. This electronic signature is analogous to a written signature. If you define a notification to require an electronic signature, users must respond to the notification from the Notification Details Web page and enter the appropriate type of signature. Otherwise, the response will not be considered valid.

- If you define a notification to require a password-based signature, users must sign their response by entering their Oracle Applications user name and password.
- If you define a notification to require a certificate-based digital signature, users must sign their response with a valid X.509 certificate issued by a certificate authority. After the signed response is submitted, Oracle Workflow performs the following steps:
  - Verifies that the signature was well formed, that it was created with a private key corresponding to the offered signing certificate, and that it is signing the plain text that it purports to sign.
  - Confirms that this user is authorized to sign the notification by checking that the certificate is assigned to a user who is a member of the recipient role for the notification.
  - Confirms that the certificate used to create the signature was valid at the time the signature was received, meaning it had not expired or been revoked. To validate a certificate, Oracle Workflow checks that the certificate does not appear on a certificate revocation list (CRL) issued by the certificate authority after the time the signature was received.

Oracle Workflow records the status of requested and submitted signatures in an evidence store, for both password-based signatures and certificate-based digital signatures. This signature information is purged when you purge obsolete workflow runtime data for the associated notifications. See: Items, *Oracle Workflow API Reference*.

The `#WF_SIG_POLICY` attribute must be either of type text or lookup. To require a password-based signature, set the value of the `#WF_SIG_POLICY` attribute to `PSIG_ONLY`. To require a certificate-based digital signature, set the value of the `#WF_SIG_POLICY` attribute to `PKCS7X509_ONLY`. If you set the value to `DEFAULT`, leave the value blank, or if you do not define a `#WF_SIG_POLICY` attribute for the message, Oracle Workflow performs the default response processing that does not require a signature.

- For ease of maintenance, you can define the `#WF_SIG_POLICY` attribute with a type of lookup and assign it the predefined Signature Policy lookup type provided in the Standard item type. The Signature Policy lookup type contains three lookup codes with the display names Password Signature, X509 Signature, and Default, representing the values `PSIG_ONLY`, `PKCS7X509_ONLY`, and `DEFAULT`, respectively.
- You can also define the `#WF_SIG_POLICY` attribute with a type of text. In this case, you must manually maintain the values that you set for this attribute.

You can either specify a constant value for the `#WF_SIG_POLICY` attribute, or specify an item type attribute as the value and include logic in your workflow process that dynamically determines at runtime whether a signature should be required or not and sets that item type attribute accordingly.

> **Note:** Electronic signatures are currently supported only for the version of Oracle Workflow embedded in Oracle Applications.

See: Setting Up for Electronic Signatures, *Oracle Workflow Administrator's Guide*, Electronic Signatures, *Oracle Workflow User's Guide*, and NtfSignRequirementsMet, *Oracle Workflow API Reference*.

#WF_SIG_ID Attribute

In Oracle Applications, you can specify an electronic signature policy for notifications by defining the `#WF_SIG_POLICY` message attribute. If you specify a signature policy that requires an electronic signature to confirm a user's response to a notification, Oracle Workflow creates another message attribute named `#WF_SIG_ID` after the notification is signed. The `#WF_SIG_ID` attribute stores the identifier for the signature, which you can use to reference information about the signature later if necessary. This attribute is of type text and has a source of 'Respond'.

> **Note:** Oracle Workflow automatically creates and sets the value of the `#WF_SIG_ID` attribute when a user submits a response to a notification with an electronic signature. You do not need to manually create or set this attribute.

#WF_SECURITY_POLICY Attribute

You can use a special message attribute with the internal name `#WF_SECURITY_POLICY` to control whether notifications that include sensitive content can be sent in e-mail. If you specify that a notification's content must not be sent in e-mail, users receive an e-mail message that only informs them that they must access the notification through the Notification Details Web page instead to view its content and respond. See: Reviewing Notifications via Electronic Mail, *Oracle Workflow User's Guide*.

The `#WF_SECURITY_POLICY` attribute must be of type text. To prevent notification content from being sent in e-mail, set the value of the `#WF_SECURITY_POLICY` attribute to `NO_EMAIL`. If you set the value to `EMAIL_OK` or `DEFAULT`, leave the value blank, or if you do not define a `#WF_SECURITY_POLICY` attribute for the message, Oracle Workflow sends the full notification content in e-mail to users whose notification preference is set to receive e-mail.

You can either specify a constant value for the `#WF_SECURITY_POLICY` attribute, or specify an item type attribute as the value and include logic in your workflow process that dynamically determines at runtime whether the notification content can be sent in e-mail or not and sets that item type attribute accordingly.

Header Attributes

In Oracle Applications, you can use special header message attributes to display key information in the notification header region of the Notification Details page. Header attributes are also displayed at the beginning of e-mail notifications. To identify a message attribute as a header attribute, you must define an internal name for it that begins with `#HDR_` using the following format:

```
#HDR_<name>
```

The display order of any header attributes in the Notification Details page or the e-mail notification is determined by the sequence of those attributes within the navigation

tree in the Workflow Builder. The display name of each header attribute will appear as a label before the attribute value in the notification header region. You can either specify a constant value for each attribute, or specify an item type attribute as the value and include logic in your workflow process that dynamically sets the item type attribute value at runtime.

> **Note:** Header attributes are currently available only for the version of Oracle Workflow embedded in Oracle Applications.

#HDR_REGION Attribute

In Oracle Applications, the notification header region by default contains general message information such as the from role, recipient, sent date, due date, and notification ID, together with any special header attributes defined for the notification and any linked Oracle Applications attachments. You can optionally use a special message attribute with the internal name `#HDR_REGION` to replace this default header with a custom header region.

Your custom header must be defined as an Oracle Applications Framework region. For more information about building an Oracle Applications Framework region, refer to Oracle*MetaLink* note 269138.1, *Oracle Applications Framework Developer's Guide*.

The `#HDR_REGION` attribute must be of type document and must have a source of Send. Set the value of this attribute to a Java Server Page (JSP) call that references your custom header region. For detailed instructions on how to specify the attribute value, see: Embedding Oracle Applications Framework Regions in Messages, page 3-38.

> **Note:** The Frame Target field is not applicable for message attributes of type document. Additionally, you must not select the Attach Content check box for the `#HDR_REGION` attribute.

> **Note:** Custom header regions are currently available only for the version of Oracle Workflow embedded in Oracle Applications.

#RELATED_APPL Attribute

In Oracle Applications, the Related Applications region in Oracle Applications Framework-based notifications by default contains icons with links to attached URLS, documents, or forms. You can optionally use a special message attribute with the internal name `#RELATED_APPL` to replace this default region with a custom Related Applications region.

Your custom Related Applications region must be defined as an Oracle Applications Framework region. For more information about building an Oracle Applications Framework region, refer to Oracle*MetaLink* note 269138.1, *Oracle Applications Framework Developer's Guide*.

The `#RELATED_APPL` attribute must be of type document and must have a source of Send. Set the value of this attribute to a Java Server Page (JSP) call that references your custom region. For detailed instructions on how to specify the attribute value, see: Embedding Oracle Applications Framework Regions in Messages, page 3-38.

> **Note:** The Frame Target field is not applicable for message attributes of type document. Additionally, you must not select the Attach Content check box for the `#RELATED_APPL` attribute.

> **Note:** Custom Related Applications regions are currently available only for the version of Oracle Workflow embedded in Oracle Applications.

#HISTORY Attribute

Response-required notifications and some FYI notifications include an action history table. You can optionally use a special message attribute with the internal name `#HISTORY` to replace the default action history table provided by Oracle Workflow with a custom action history region. See: Action History, page 3-24.

Your custom action history table must be defined as an Oracle Applications Framework region, or as a PL/SQL or PL/SQL CLOB document that contains text or HTML. For more information about building an Oracle Applications Framework region, refer to Oracle*MetaLink* note 269138.1, *Oracle Applications Framework Developer's Guide*.

The `#HISTORY` attribute must be of type document and must have a source of Send. For an Oracle Applications Framework region, set the value of this attribute to a Java Server Page (JSP) call that references your custom action history region. For detailed instructions on how to specify the attribute value, see: Embedding Oracle Applications Framework Regions in Messages, page 3-38.

For a PL/SQL or PL/SQL CLOB document, set the attribute value to the procedure and document ID from which the document is generated. See: To Define a Document Attribute, page 3-13.

> **Note:** The Frame Target field is not applicable for message attributes of type document. Additionally, you must not select the Attach Content check box for the `#HISTORY` attribute.

#SUBMIT_COMMENTS Attribute

The action history region for notifications begins with a row for the initial submission of the process, when the notification is first sent. You can optionally use a special message attribute with the internal name `#SUBMIT_COMMENTS` to display comments from the process owner for this initial action. See: Action History, page 3-24.

The `#SUBMIT_COMMENTS` attribute must be of type text and must have a source of Send. You should specify an item type attribute as the value for the `#SUBMIT_COMMENTS` attribute and include logic in your workflow process that dynamically sets that item type attribute to contain the comments at runtime. Ensure that you provide a way for the process owner to enter the comments when submitting the process and store the comments in the appropriate item type attribute.

Region Personalization Attributes

In Oracle Applications, you can use special message attributes to apply personalizations at different levels to Oracle Applications Framework regions embedded in notifications.

- `#PERZ_FUNCTION_NAME` - Define this attribute for a message to apply a personalization at function level to a region embedded in the notification. Set the attribute value to the function name.
- `#PERZ_ORGANIZATION_ID` - Define this attribute for a message to apply a personalization at organization level to a region embedded in the notification. Set the attribute value to the organization ID.

- #PERZ_LOCALIZATION_CODE - Define this attribute for a message to apply a personalization at localization level to a region embedded in the notification. Set the attribute value to the localization code.

You can also create personalizations for an Oracle Applications Framework region at site level. In this case the personalizations will automatically be applied when the region appears embedded in a notification, as well as when it appears in any other location.

Notification Mailer Attributes

You can use special message attributes to control how a notification mailer generates the e-mail message for a notification, if the recipient has a notification preference to receive e-mail notifications. For example, if you want to customize notifications from a particular department, you can define these attributes for those notifications.

- #WFM_FROM - Define this attribute for a message to specify the value that appears in the From field of the message header when the e-mail notification message is delivered to a user. If the #WFM_FROM message attribute is defined for a notification, the notification mailer that sends the message will use the #WFM_FROM attribute value in the From field for that message, overriding the mailer's From Address parameter value.
- #WFM_REPLYTO - Define this attribute for a message to specify the address of the e-mail account that receives incoming messages, to the which e-mail notification response should be sent. If the #WFM_REPLYTO message attribute is defined for a notification, the notification mailer that sends the message will use the #WFM_REPLYTO attribute value as the reply address for that message, overriding the mailer's Reply To parameter value.
- #WFM_HTMLAGENT - Define this attribute for a message to specify the base URL that identifies the HTML Web agent that handles HTML notification responses. This URL is required to support e-mail notifications with HTML attachments. The default URL is derived from the Workflow Web Agent specified in the Global Preferences Web page in standalone Oracle Workflow, or from the Applications Web Agent (APPS_WEB_AGENT) profile option in Oracle Applications. However, you can override this default by defining a different value for this attribute. If the #WFM_HTMLAGENT message attribute is defined for a notification, the notification mailer that sends the message will use the #WFM_HTMLAGENT attribute value as the HTML Web agent for that message, overriding the mailer's HTML Agent parameter value.
- #WFM_LANGUAGE - Define this attribute for a message to specify the value of the NLS_LANGUAGE database initialization parameter that determines the language for the e-mail notification. Set the attribute value to a language name supported by the Oracle Database, such as ARABIC. If the #WFM_LANGUAGE message attribute is defined with a valid value for a notification, the notification mailer that sends the message will use the #WFM_LANGUAGE attribute value when generating the e-mail message, overriding the language preference of the recipient role.
- #WFM_TERRITORY - Define this attribute for a message to specify the value of the NLS_TERRITORY database initialization parameter that determines the territory-dependent formatting for the e-mail notification. Set the attribute value to a territory name supported by the Oracle Database, such as UNITED ARAB EMIRATES. If the #WFM_TERRITORY message attribute is defined with a valid value for a notification, the notification mailer that sends the message will use the

#WFM_TERRITORY attribute value when generating the e-mail message, overriding the territory preference of the recipient role.

> **Note:** The #WFM_LANGUAGE and #WFM_TERRITORY attributes function independently of each other. If you define only one of these attributes with an overriding value for the corresponding NLS parameter, the notification mailer still uses the recipient role's preference for the other parameter.

For more information, see the Oracle Workflow Manager online help and Locale Data, *Oracle Database Globalization Support Guide*.

Notification Mailer Message Template Attributes

You can use special message attributes to specify which message templates a notification mailer uses to generate the e-mail message for a notification, if the recipient has a notification preference to receive e-mail notifications. For example, if you want to customize notifications from a particular department, you can define these attributes for those notifications. The templates specified in these attributes for a particular notification override the templates specified in the configuration parameters for a notification mailer.

You can create your own message templates by using the Workflow Builder to define skeleton messages within an item type. See: Modifying Your Message Templates, *Oracle Workflow Administrator's Guide*.

The value for a message template attribute must be specified in the following format:

```
<item_type_internal_name>:<message_internal_name>
```

For example, to specify the Workflow Open Mail (Templated) message within the System: Mailer item type, you would enter the following value:

```
WFMAIL:OPEN_MAIL
```

You can either specify a constant value for a message template attribute, or specify an item type attribute as the value and include logic in your workflow process that dynamically determines at runtime which message template to use and sets that item type attribute accordingly.

You can define the following attributes to specify templates to be used for a notification message at various stages in e-mail notification processing.

- #WFM_OPEN_MAIL - Specify the template to use for e-mail notifications that require a response, if you are using the templated response method.
- #WFM_OPEN_MAIL_DIRECT - Specify the template to use for e-mail notifications that require a response, if you are using the direct response method.
- #WFM_OPEN_MAIL_FYI - Specify the template to use for e-mail notifications that do not require a response.
- #ATTACHED_URLS - Specify the template to use to create the Notification References attachment for HTML-formatted notification messages that include URL attributes with Attach Content checked.
- #WFM_CANCELED - Specify the template to use to inform the recipient that a previously sent notification is canceled.

- `#WFM_OPEN_INVALID` - Specify the template to send to a user when the user responds incorrectly to a notification.
- `#WFM_CLOSED` - Specify the template to use to inform the recipient that a previously sent notification is now closed.
- `#WFM_OPEN_MORE_INFO` - Specify the template to use to send a request for more information from one user to another user.

For more information, see the Oracle Workflow Manager online help.

Example 'Respond' Message Attributes

Following are examples of how the Notification System generates the Response section of an e-mail notification using a sample set of 'Respond' message attributes that have no default values.

The following table lists some sample 'Respond' message attributes.

Sample Respond Attributes

Internal Name	Type	Format/Lookup Type	Display Name	Description
RESULT	lookup	WFSTD_AP PROVAL	Action	Do you approve?
COMMENT	text	2000	Review Comments	
REQDATE	date	DD-MON-YYYY	Required Date	If there is no required date, leave this blank.
MAXAMT	number		Maximum Amount	This is the maximum approved amount.

For the templated response method, the following boilerplate text is used to generate the Response template section of an e-mail notification:

```
<Description>
<Display Name>: " "
          <list of lookup codes>
```

Portion of Resulting Response Template as Shown in a Templated Response E-mail Notification

Do you approve?

Action: " "

Approve

Reject

Review Comments: " "

If there is no required date, leave this blank.

Required Date: " "

This is the maximum approved amount.

Maximum Amount: " "

For the direct response method, the following boilerplate text is used to generate the Response section of an e-mail notification:

`Enter the <Display Name> on line <Sequence>. <Description> <Type_Hint>`
`<Display Name>` is replaced with the Display Name of the message attribute. `<Sequence>` is replaced with the relative sequence number of the 'Respond' message attribute as it appears in the Navigator tree among all 'Respond' message attributes (that is, the presence of 'Send' message attributes is ignored when determining the sequence). `<Description>` is replaced with the Description of the message attribute. In addition, `<Type_Hint>` is replaced with a hint statement if the message attribute matches a data type that has a hint. The following table shows the data types with hints and the corresponding hint statements.

Data Type Hints

Type	Type Hint
Lookup	Value must be one of the following: <*list of lookup codes*>
Date	Value must be a date [in the form "<*format*>"].
Number	Value must be a number [in the form "<*format*>"].
Text	Value must be <*format*> bytes or less.

Portion of Resulting Response Section as Shown in a Direct Response E-mail Notification

Enter the Action on line 1. Do you approve? Value must be one of the following:

Approve

Reject

Enter the Review Comments on line 2. Value must be 2000 bytes or less.

Enter the Required Date on line 3. If there is no required date, leave this blank. Value must be a date in the form "DD-MON-YYYY".

Enter the Maximum Amount on line 4. This is the maximum approved amount. Value must be a number.

Embedding Oracle Applications Framework Regions in Messages

If you are using Oracle Applications and you have Oracle Applications Framework set up in Oracle JDeveloper for custom development, you can embed Oracle Applications Framework regions in a notification message. To embed a region, define a message attribute whose value is a Java Server Page (JSP) call that references the region.

The message attribute representing the region must be of type document and must have a source of Send. You can assign the attribute any appropriate internal name, display name, and description.

Specify the value for the attribute in the following format:

```
JSP:/OA_HTML/OA.jsp?OAFunc=<Function_Name>
```

where *<Function_Name>* is the self-service function name for the region you want to embed in the notification, as registered in the Form Functions window in Oracle Applications. For example:

```
JSP:/OA_HTML/OA.jsp?OAFunc=EMBED_WL_FUNC
```

To pass parameters to the region, specify the value for the attribute in the following format:

```
JSP:/OA_HTML/OA.jsp?OAFunc=<Function_Name>&<Name1>=<Value1>
&<Name2>=<Value2>...
```

where *<Function_Name>* is the function name for the region, *<Name1>* and *<Value1>* are the parameter name and value for the first parameter, *<Name2>* and *<Value2>* are the parameter name and value for the second parameter, and so on. For example:

```
JSP:/OA_HTML/OA.jsp?OAFunc=EMBED_WL_FUNC&Order_Type=12345
```

You can set the value for the message attribute in the following ways:

- Specify the complete JSP call as a constant default value for the message attribute.

- Specify an item type attribute of type document as the message attribute value, and include logic in your workflow process that dynamically sets the item type attribute value at runtime. This method lets you dynamically set the complete JSP call.
- Use token substitution in the message attribute value to dynamically set the function name within the JSP call.
  - Create an item type attribute for the function name.
  - Create an additional message attribute for the function name, and specify the corresponding item type attribute as the value of that message attribute.
  - Include logic in your workflow process that dynamically sets the item type attribute value to the appropriate function name at runtime.
  - Specify the value for the message attribute representing the region in the following format:

    `JSP:/OA_HTML/OA.jsp?OAFunc=-&funcname_attr-`

    where *funcname_attr* is the internal name of the message attribute that specifies the function name.
- Use token substitution in the message attribute value to dynamically set the parameter values for the function within the JSP call. In this case you must specify the function name and the parameter names as constants within the JSP call, and token substitute only the parameter values.
  - Create an item type attribute for each parameter value.
  - Create an additional message attribute for each parameter value, and specify the corresponding item type attribute as the value of that message attribute.
  - Include logic in your workflow process that dynamically sets the item type attribute values to the appropriate parameter values at runtime.
  - Specify the value for the message attribute representing the region in the following format:

    `JSP:/OA_HTML/OA.jsp?OAFunc=<Function_Name>&<Name1>=-&msgattr1-&<Name2>=-&msgattr2-...`

    where *<Function_Name>* is the function name for the region, *<Name1>* and *<Name2>* are the first and second parameter names, *msgattr1* and *msgattr2* are the internal names of the message attributes that specify the first and second parameter values, and so on.

    Note: The Frame Target field is not applicable for message attributes of type document. Additionally, you must not select the Attach Content check box for a message attribute that references a region. An Oracle Applications Framework region can only be displayed within the message body of a notification. It cannot be included as an attachment to the notification.

To embed the region in the message, enter the token for the message attribute in the HTML body for the message. A message can include multiple regions, which will be displayed in the sequence in which their tokens appear in the HTML Body field. Additionally, the message can include calls to the special message function `WF_NOTIFICATION()` to produce a table of message attributes or an

action history table, which will also be rendered as Oracle Applications Framework regions. See: WF_NOTIFICATION() Message Function, page 3-26.

> **Note:** If you embed an Oracle Applications Framework region in a message, then the message body can only contain message attribute tokens referencing such regions and calls to the special message function `WF_NOTIFICATION()`. The message body cannot contain any tokens for message attributes that do not reference regions.

Leave the Text Body field blank for an Oracle Applications Framework region message. Oracle Workflow does not use the text body for such messages. Instead, a text version of the message is derived directly from the included regions. Note that non-text content such as images, links, or special HTML formatting will not appear in the text version of a region.

Follow these guidelines when you are developing an Oracle Applications Framework region for inclusion in a notification message:

- Oracle Workflow notifications support embedded Oracle Applications Framework regions only. You cannot include an entire Oracle Applications Framework page.
- The region style must be Stack Layout. This region style lets users personalize the region with Oracle Applications Framework Personalization.
- Ensure that the event handling in the controllers for the embedded Oracle Applications Framework region does not interfere with the event handling of the region controller for the main message body. To avoid such interference, when handling any events in the controller code, you must qualify those events with the region associated with the controller.
- The embedded region must be complete within itself. You should associate the region with its own application module and assign it a region title.
- The embedded region must be a view-only region. That is, it cannot contain elements that allow user input, such as buttons.
- The embedded region must not contain any information that is specific to a particular user session.
- If you do not want to send an embedded region in e-mail, you can exclude the message body from notification e-mail messages by assigning the notification a message template that directs recipients to access the notification through the Worklist Web pages instead.
  - For a notification that requires a response, define the special message attributes `#WFM_OPEN_MAIL` and `#WFM_OPEN_MAIL_DIRECT`, and set the values of these attributes to `WFMAIL:VIEW_FROMUI` to use the Workflow View From UI message template.
  - For a notification that does not require a response, define the special message attribute `#WFM_OPEN_MAIL_FYI`, and set the value of this attribute to `WFMAIL:VIEW_FROMUI_FYI` to use the Workflow View FYI From UI message template.

  See: Notification Mailer Message Template Attributes, page 3-35, Workflow View From UI Message, *Oracle Workflow Administrator's Guide*, and Workflow View FYI From UI Message, *Oracle Workflow Administrator's Guide*.

- If the embedded region includes subtabs or Hide/Show regions, ensure that you provide alternate representations of that content for inclusion in an e-mail message or in the plain text version of a notification used for electronic signatures. You can choose to exclude the message body from e-mail, or you can code alternate representations in the controllers for the embedded region.

  For example, for a subtab region you can create a separate region displaying that content and include the separate region programmatically in the e-mail or plain text cases. For Hide/Show regions, you should remove the Hide/Show region and instead display all the child rows of the Hide/Show region directly in the current container.

- For usability reasons, we recommend that if your Stack Layout region includes a Table Layout region, you should restrict to 200 the number of rows returned from the query for the Table Layout region.

  If your Stack Layout region includes a region in the Table Layout region style, the Notification Details page displays the Table Layout region with the Next and Previous rowset functionality available when appropriate. However, because e-mail clients cannot interpret the rowset functionality, e-mail notifications display the Table Layout region with static row data, which is restricted to 200 rows.

- Do not enable sorting or navigation features for table components programmatically at runtime.

- Register your Oracle Applications Framework region as a self-service function (non-form function) in the Form Functions window in Oracle Applications. The internal name that you define for the function is the function name that you must include in the message attribute value to reference the region. See: Form Functions Window, *Oracle Applications Developer's Guide*.

  When you register the function, enter the HTML Call in the Web HTML tabbed region of the Form Functions window, using the following format:

  ```
  OA.jsp?page=/<directory_path>/<Region_Code>
  &akRegionApplicationId=<Region_Application_ID>
  ```

  For example:

  ```
  OA.jsp?page=/oracle/apps/fnd/wf/worklist/webui/
  WFNTFWORKLIST&akRegionApplicationId=601
  ```

For more information about building an Oracle Applications Framework region, refer to Oracle*MetaLink* note 269138.1, *Oracle Applications Framework Developer's Guide*.

Related Topics

To Create a Message, page 3-42

To Define a Message Attribute, page 3-46

To View the Details of a Notification (for Oracle Applications), *Oracle Workflow User's Guide*

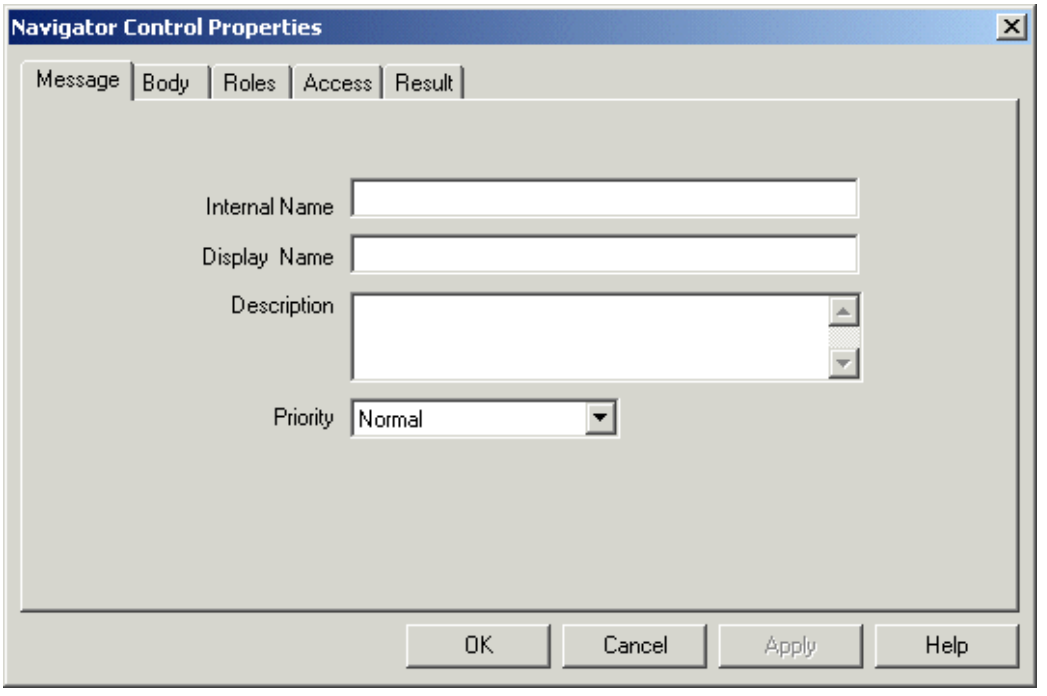
Reviewing Notifications via Electronic Mail, *Oracle Workflow User's Guide*

Defining Messages

To Create a Message:

1. Select the item type that you want to create a message for in the navigator tree, and choose New Message from the Edit menu. A Message property page appears.

Message Property Page



2. Provide an internal name for the message that is all uppercase with no leading/trailing spaces, and provide a display name. You may also enter an optional description. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying a message.

   Important: To update the internal name for a message once it is defined, you must use a special SQL script called `wfchmsg.sql`. You should only use this script to correct errors in a message's internal name during design time. Do not use this script to rename messages that are involved in running instances of processes. See: Wfchmsg.sql, *Oracle Workflow Administrator's Guide*.

   Caution: Do not include colons "`:`" or leading/trailing spaces in your internal name.

3. Choose High, Normal, or Low for the default priority of the message. The priority level simply informs the recipient of the urgency of the message. It does not affect the processing or delivery of the message.

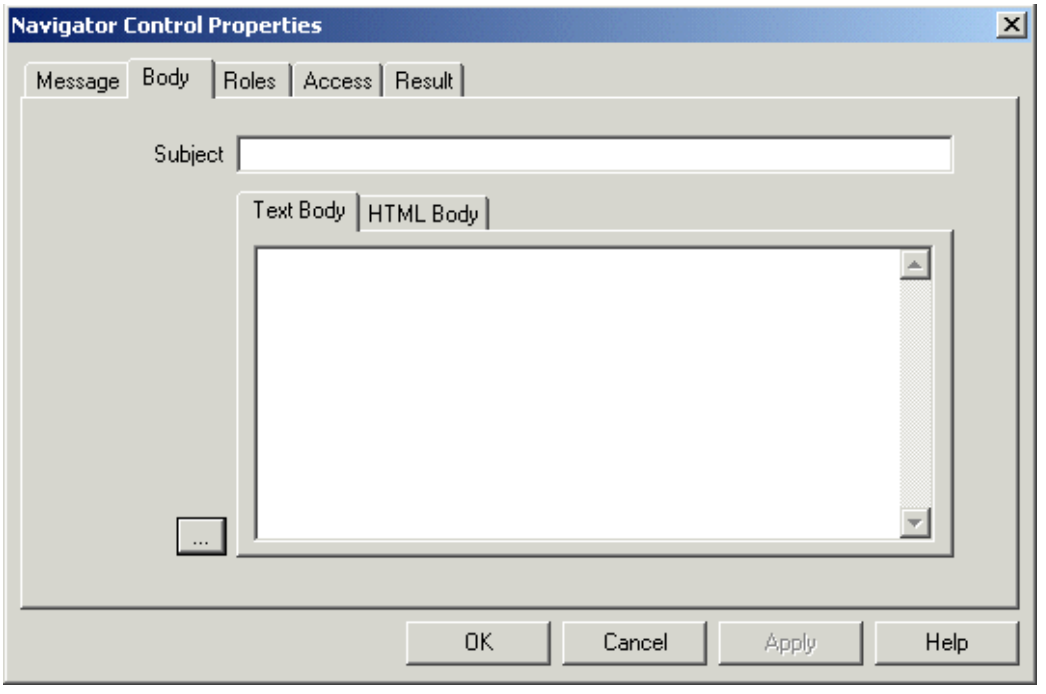
   Note: When you assign this message to a notification activity and you incorporate the notification activity into a process diagram as a node, you can override this default message priority with

a new priority that is constant or dynamically determined at runtime. See: To Define Nodes in a Process, page 4-4.

Note: In earlier versions of Oracle Workflow, the message priority was represented as a numeric value between 1 (high) and 99 (low). Oracle Workflow now automatically converts the priority values of all message definitions defined in earlier versions as follows: 1-33 = High, 34-66=Normal, and 67-99=Low.

4. Choose Apply to save your changes.
5. Select the Body tab to display the Body property page of the message.

Body Property Page



6. The subject gets its default value from the display name that you entered in the Message tab. You can choose to keep the default subject or enter a new subject for the message. The subject can include message attributes that get token replaced with runtime values when the message is delivered. To include a message attribute in the subject, use an ampersand (`&`) followed by the message attribute's internal name. See: Send and Respond Message Attributes, page 3-23 and To Define a Message Attribute, page 3-46.

   Tip: For clarity, you can assign a message attribute the same name as the item type attribute it references.

7. If the message will not contain any embedded Oracle Applications Framework regions, enter a plain text message body in the Text Body field. You can select the ellipsis button (`...`) to expand the view of the Subject and Text Body fields in another window.

Oracle Workflow uses the content you enter in the Text Body field to generate a plain text version of the notification message. The plain text message can be viewed from the from an e-mail reader that displays plain text messages.

> **Important:** If the message will not contain any embedded Oracle Applications Framework regions, ensure that you enter a plain text message body in the Text Body field. If Text Body is null, you get an empty notification when you view your message from a plain text e-mail reader. However, if the message does contain one or more embedded Oracle Applications Framework regions, then you should leave the Text Body field blank, because the text body is not used for this type of notification. See: Embedding Oracle Applications Framework Regions in Messages, page 3-38.

8. Enter an HTML-formatted message body in the HTML Body field by selecting the HTML Body tab and typing in the content, or by selecting Import to import the content from a `.HTM` or `.HTML` file. You can also select the ellipsis button (`...`) to expand the view of the Subject and HTML Body fields in another window.

   If the message will contain an embedded Oracle Applications Framework region, then the HTML body is required. Otherwise, the HTML Body field is optional. See: Embedding Oracle Applications Framework Regions in Messages, page 3-38.

   > **Important:** When you enter or import the HTML message body, you do not need to include the `<Body>...</Body>` HTML tags. If you do include these tags, Oracle Workflow simply extracts the content between these tags to use as the HTML message body. As a result, Oracle Workflow ignores any HTML tags or content prior to the `<Body>` tag.

   > **Important:** Oracle Workflow Builder does not verify the HTML formatting of the message body.

   Oracle Workflow uses the content you enter in the HTML Body field to generate an HTML-formatted version of the notification message. You can view an HTML-formatted notification message from the Notification Details Web page, or from an e-mail reader that displays HTML-formatted messages or HTML-formatted message attachments.

   > **Note:** If HTML Body is null, Oracle Workflow uses the message body entered in Text Body to generate the notification message. It inserts the plain text between the `<pre>...</pre>` HTML tags.

9. You can embed message attributes in the text or HTML body. Oracle Workflow token replaces the message attributes with runtime values when it delivers the notification. To embed a message attribute, enter an ampersand ("`&`") followed by the message attribute's internal name.

   > **Important:** The text in a message body must be less than 4000 bytes. If you include message attributes in the text for token substitution, then the final message body can increase up to 32000 bytes.

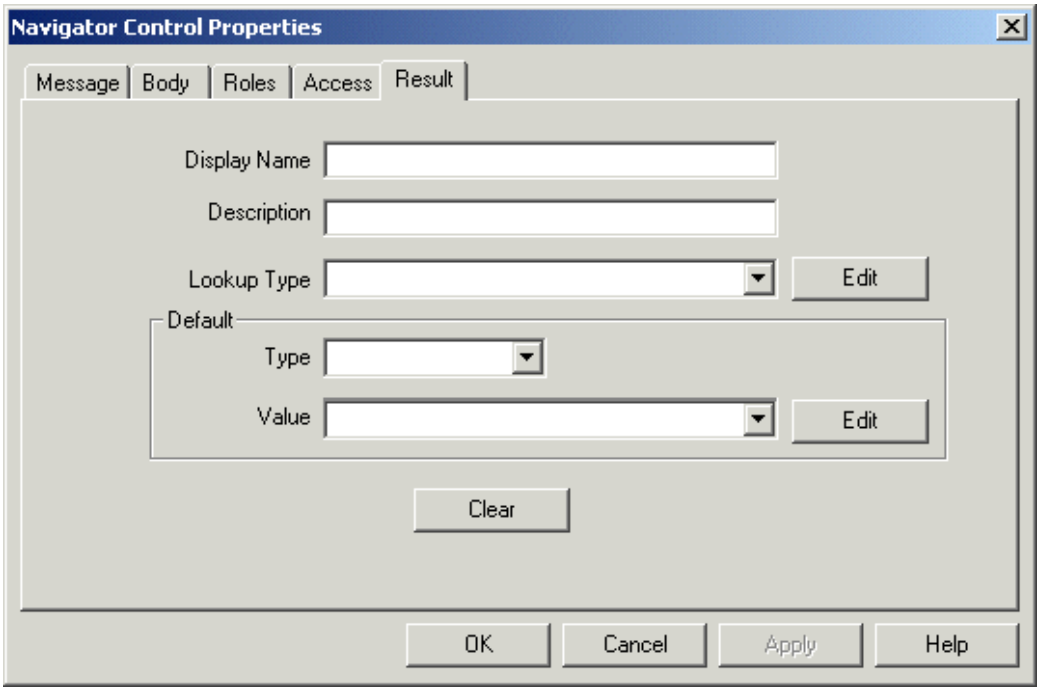
Note: You can also include a special token in the message subject or body called `&#NID`. Oracle Workflow substitutes this token with the notification ID of the runtime notification.

Note: If you define a 'Send' message attribute of type URL that points to an image file with an extension of `gif`, `jpg`, `png`, `tif`, `bmp`, or `jpeg`, and you embed the URL attribute in the HTML body of the message, then Oracle Workflow displays the image inline in the Notification Details page and HTML-formatted e-mail notifications. See: To Define a URL Attribute, page 3-10.

Additionally, you can use the message function `WF_NOTIFICATION()` to include a formatted table of message attributes or an action history table in the text or HTML message body.

10. Choose Apply to save your changes.
11. Select the Roles tab page to specify the roles that have access to this message. (This functionality will be supported in a future release.)
12. Select the Access tab page to set the access levels allowed to modify this message. See: Allowing Access to an Object, page 3-17.
13. If you want the notification message to prompt the performer for a response value and you want Oracle Workflow to interpret that response value as the result of the notification activity, select the Result tab page and complete the information requested. Oracle Workflow uses the information you specify in the Result tab page to create a special 'Respond' message attribute called `RESULT`. See: Message Result, page 3-23.

Result Property Page



Specify a display name and description for `RESULT`. Select a lookup type from the poplist field. The lookup type you select should be identical to the lookup type specified for the notification activity's result type. Select a lookup code in the Default Value region. The lookup code you select appears as the default value of the `RESULT` message attribute.

> **Note:** To create any other type of message attribute, see: To Define a Message Attribute, page 3-46.

14. Choose Apply to save your changes, OK to save your changes and close the property page, or Cancel to cancel your changes and close the property page.
15. The message you just defined now appears beneath the Message branch in the navigator tree. You can review or edit the properties of this message at any time by double-clicking on the message in the navigator tree or by selecting the message and choosing Properties from the Edit menu.

    If a message has a Result defined, then its message icon in the Navigator tree has a red question mark overlay to help you distinguish it from messages that do not have a Result defined.
16. You must now define all the message attributes that you have included in the subject and body of this message.
17. To create a message attribute that references an item type attribute, select the referenced item type attribute in the navigator tree, and hold down your mouse select button as you drag the item type attribute to your message.

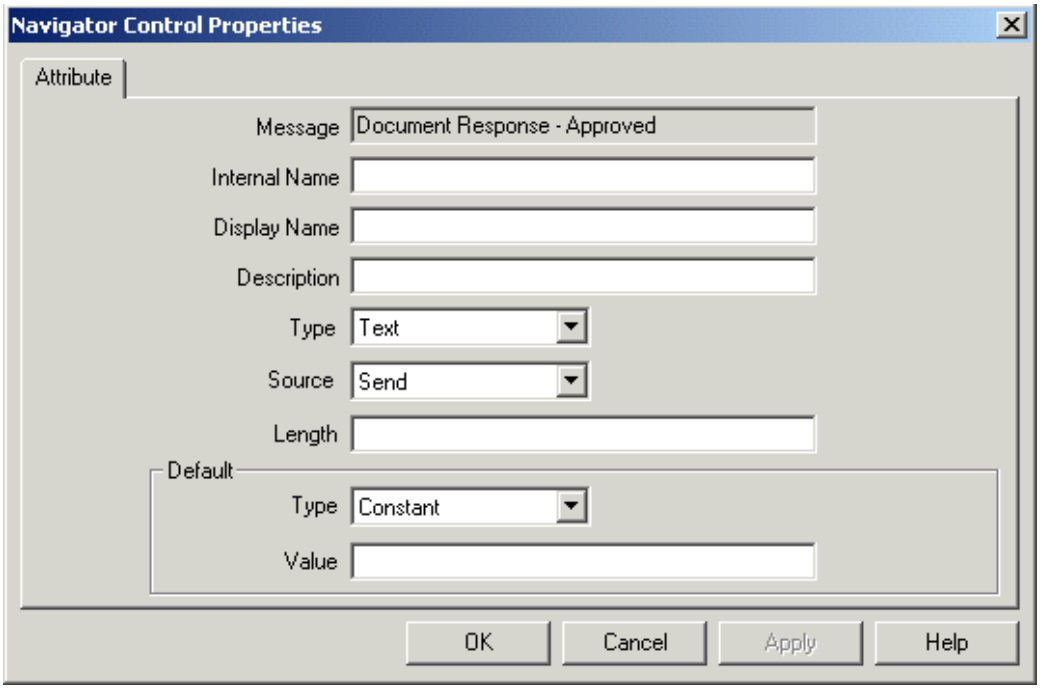
    Edit the property page that appears, making sure the message attribute has the proper Source. The Default Value region is automatically set to Item Attribute and references the originating item attribute.
18. You can also create message attributes that are not based on existing item type attributes. See: To Define a Message Attribute, page 3-46.

To Define a Message Attribute:

1. To create a message attribute that does not reference an existing item type attribute, select a message in the navigator tree and choose New Attribute from the Edit menu.

   An Attribute property page appears.

Message Attribute Property Page



2. Provide an Internal Name in all uppercase with no leading/trailing spaces. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying an attribute.

   Important: To update the internal name for a message attribute once it is defined, you must use a special SQL script called `wfchmsga.sql`. You should only use this script to correct errors in a message attribute's internal name during design time. Do not use this script to rename message attributes that are involved in running instances of processes. See: Wfchmsga.sql, *Oracle Workflow Administrator's Guide*.

   Caution: Do not include leading/trailing spaces or reserved characters such as a colon (":"), ampersand ("&"), or number sign ("#") in the internal name of a message attribute. Additionally, do not include any spaces within the internal name of a message attribute.

3. Specify 'Send' or 'Respond' in the Source field to indicate whether this attribute should send information to the notification recipient or prompt a notification message recipient for a response, respectively.

4. Enter a Display Name. This is the name that appears in the navigator tree. If this is a 'Respond' message attribute, then this display name is also used as the response prompt. See: Example 'Respond' Message Attributes, page 3-36.

   For 'Send' message attributes, the Display Name of the attribute appears in plain text e-mail notifications only if the attribute is of type URL, to describe what the URL drills down to.

5. Enter an optional description. If this is a 'Respond' message attribute, use this field to elaborate on response instructions. See: Example 'Respond' Message Attributes, page 3-36.
6. Select the data type of the attribute.
7. Depending on the Type of your attribute, provide the following default information:
   - **Text** - Specify the maximum length of the text attribute.
   - **Number** - Optionally provide a format mask for your number and a default number value.
   - **Date** - Optionally supply a format mask for the date and a default date value.
   - **Lookup** - Choose the name of a predefined Lookup Type from which to draw values. Choose a lookup code for the default value.
   - **URL** - Specify a Universal Resource Locator (URL) to a network location in the Default Value field. See: To Define a URL Attribute, page 3-10.

     Important: 'Respond' message attributes of type URL do not appear properly when you view the notification from a plain text e-mail reader. You should advise your workflow users to view their notifications from the Notification Details Web page if you plan to create messages with 'Respond' message attributes of type URL.

     Important: A single 'Respond' message attribute of type URL replaces the Notification Details Web page response frame and takes the notification recipient to a custom HTML page to complete the notification response. Your custom HTML response page must include references to all your 'Respond' message attributes, including the special `RESULT` attribute, if one is defined, and must also include a call to the Workflow Engine *CompleteActivity()* API to inform the Workflow Engine when the notification response is complete.

     Note: If you define a 'Send' message attribute of type URL that points to an image file with an extension of `gif`, `jpg`, `png`, `tif`, `bmp`, or `jpeg`, and you embed the URL attribute in the HTML body of the message, then Oracle Workflow displays the image inline in the Notification Details page and HTML-formatted e-mail notifications. See: To Define a URL Attribute, page 3-10.

   - **Form** - This attribute is relevant only with the version of Oracle Workflow embedded in Oracle Applications.

     Specify a developer form function name and any optional argument string (form function parameters) in the Default Value field. See: Overview of Menus and Function Security, *Oracle Applications Developer's Guide* and To Define a Form Attribute, page 3-12.

     Important: 'Send' and 'Respond' message attributes of type Form appear only when your Worklist Web pages are launched

from Oracle Applications. The attached form icon is enabled if a notification message includes a 'Send' message attribute of type Form. The notification recipient can click on the attached form icon to drill down to the form function defined by the message attribute.

Important: If a message includes a 'Respond' message attribute of type Form, the attached form icon that appears in the notification's Response section simply drills down directly to the designated form function. This form function must be coded with a call to the Workflow Engine *CompleteActivity()* API to inform the Workflow Engine that the notification response is complete. See: Workflow Engine APIs, *Oracle Workflow API Reference*.

- **Document** - Enter a string that identifies the document in the Default Value field. See: To Define a Document Attribute, page 3-13.
- **Role** - Specify a role name. If a message attribute of type role is included in a notification message, the attribute automatically resolves to the role's display name, eliminating the need for you to maintain separate attributes for the role's internal and display names. Also when you view the notification from a Web browser, the role display name is a hypertext link to the e-mail address for that role. To set a default value for the attribute, you must initially load roles from the database. See: Roles, page 4-19.
- **Event** - Specify an event item type attribute as the default value.

  Important: Do not specify a message attribute's data type as Attribute, as it serves no purpose in a notification message and is also not supported by the Workflow Notification System.

  Important: 'Respond' message attributes of type Date, Number, Text, Document or Role prompt the notification recipient to respond with a date, number, text value, document, role (internal or display name), respectively.

  'Respond' message attributes of type Lookup prompt the notification recipient to select a response from a list of values.

8. If your message attribute is a URL attribute, specify a Frame Target. When you reference this message attribute in a message and the user who receives the message clicks the URL link in the message body, the URL opens according to what you specify as the frame target. The frame target can be:
   - New Window - the URL loads in a new, unnamed browser window.
   - Same Frame - the URL loads in the same frame as the element that references the URL attribute.
   - Parent Frameset - the URL loads into the immediate FRAMESET parent of the current frame. This value is equivalent to Same Frame if the current frame has no parent.

- Full Window - the URL loads into the full, original window, thus cancelling all other frames. This value is equivalent to Same Frame if the current frame has no parent.

  Note: In Oracle Applications, you should only choose New Window or Full Window as the frame target for URL attributes. In the Oracle Applications version of the Worklist, the Notification Details page does not use frames to display notifications. Consequently, choosing Same Frame or Parent Frameset is equivalent to choosing Full Window in this case.

  Note: If the URL attribute points to an image file with an extension of `gif`, `jpg`, `png`, `tif`, `bmp`, or `jpeg`, and you embed the URL attribute in the notification message by token substitution, then Oracle Workflow displays the image inline in the Notification Details page and HTML-formatted e-mail notifications, rather than as a link. In this case the frame target value is not applicable.

9. If your message attribute is a Send attribute, is of type Document, and contains a PL/SQL, PL/SQL CLOB, or PL/SQL BLOB document, you can check Attach Content to attach the content of the attribute to the notification message. When you view your notification from the Notification Details Web page, you see a document icon following the notification message body that displays the contents of the attached message attribute when you click on it. If you view your notification from e-mail, the presentation of the attachment will vary depending on what your e-mail notification preference setting is. See: Reviewing Notifications via Electronic Mail, *Oracle Workflow User's Guide*.

   Note: You can attach, as well as embed (by token substitution) PL/SQL document attributes in the notification message and are not limited to one or the other. However, a document attribute that references an Oracle Applications Framework region can only be embedded in the notification message. A region cannot be attached to a notification. See: Embedding Oracle Applications Framework Regions in Messages, page 3-38.

10. If your message attribute is a Send attribute and is of type URL, you can check Attach Content to attach the URL to the message.

    - In the Notification Details Web page, an attached URL appears with an attachment icon after the notification message body.
    - For a notification e-mail message that does not include an embedded Oracle Applications Framework region, Oracle Workflow appends an attachment called Notification References to the e-mail. This attachment includes a link to each URL attribute for the message that has Attach Content checked. You can navigate to a URL by choosing its link.
    - For a notification e-mail message that includes an embedded Oracle Applications Framework region, Oracle Workflow includes a Related Applications region in the e-mail after the message body. The Related Applications region displays a link to each attached URL attribute.

Note: You can attach, as well as embed (by token substitution) URL attributes in the notification message and are not limited to one or the other.

Note: If you check Attach Content for a URL attribute that points to an image file, the URL attribute appears as a link in the Notification References attachment or Related Applications region, just as other URLs do.

11. For message attributes, the default value may be a constant or an item type attribute. If the default references its entire value directly from an item type attribute, choose Item Attribute, then use the poplist field to choose an item type attribute. The item type attribute you select must be associated with the same item type that the message itself is associated with. The item type attribute you select must also be of the same data type as the message attribute.

    Note: A message attribute type of 'Text' is compatible with any item attribute type except Event, but all other message attribute types must match the item attribute type exactly.

12. Choose Apply to save your changes, OK to save your changes and close the property page, or Cancel to cancel your changes and close the property page.
13. Any message attribute you define appears beneath the respective message you defined it for in the navigator tree. You can review or edit the properties of an attribute at any time by double-clicking on the attribute in the navigator tree or by selecting the attribute and choosing Properties from the Edit menu. Respond message attribute icons in the Navigator tree have a red question mark overlay to help you distinguish them from Send message attribute icons.

    Note: Message attributes assume the access/protection level of their parent message.

    Important: The order in which you list 'Respond' message attributes in the navigator tree correlates to the order in which they appear in the response section of the notification message. You can use the drag and drop feature of the navigator tree to reorder a set of attributes, or select an attribute and choose Move Attribute Up or Move Attribute Down from the Edit menu.

To Token Substitute an Attribute:

Oracle Workflow supports runtime token substitution of attributes. You can embed attributes within an attribute as well as embed attributes within a message subject and body. To embed an attribute, specify the attribute that you want to have token substituted as `&attr_name`, where `attr_name` is the internal name of the attribute.

When performing token substitution, Oracle Workflow fetches the internal name of the attribute and its value. If an attribute requiring token substitution is nested with another attribute, Oracle Workflow orders the nested list of attributes according to the length of their internal attribute names and then begins substituting the attributes with the longest internal names first.

Important: If you find that you need to nest message attributes more than two layers deep to display the necessary message body content, you should investigate creating a PL/SQL document-type message attribute. See: External Document Integration, page 3-5.

To Copy a Message:

1. Select the message to copy in the navigator tree.
2. Hold down your mouse select button as you drag the message to the item type branch you want to copy to.
3. When you release your mouse button, a property page appears for the new message.

   Note: You can also use the Copy and Paste options in the Edit menu.

4. Enter a new internal name and display name.
5. Make any additional modifications to the properties of the message.
6. When you are done, choose OK.

   Note: Copying a message also copies any message attributes assigned to it.

Related Topics

Using the Edit Button in a Property Page, page 2-6

To Copy an Attribute, page 3-17

Reviewing Notifications via Electronic Mail, *Oracle Workflow User's Guide*

Activities

An activity is a unit of work that contributes toward the accomplishment of a process. An activity can be a notification, a function, an event, or a process. A notification activity sends a message to a workflow user. The message may simply provide the user with information or request the user to take some action. A function activity calls a PL/SQL stored procedure or some external program to perform an automated function. An event activity receives, raises, or sends a business event. A process activity is a modelled workflow process, which can be included as an activity in another process to represent a sub-process.

Activities are organized beneath their respective Processes, Notifications, Functions, or Events headings in the navigator tree. You can create, edit, and delete activity definitions in the navigator tree, and drag an activity from the tree into a Process window to create a new usage of that activity in a process diagram. Each activity is depicted as an icon in a process diagram

Oracle Workflow provides an item type called Standard that includes generic activities you can use in any process you define. For example, some of the activities perform standard functions such as comparing two values. See: Standard Activities, page 5-1.

Oracle Workflow also provides an item type called System:Error that includes standard error processes and activities you can use to create a custom error process. You can assign an error process to a process activity. If an error occurs, the error process informs

Oracle Workflow how to handle the error. See: Error Handling for Workflow Processes, page 11-1.

Notification Activity

When the workflow engine reaches a notification activity, it issues a Send() API call to the Notification System to send the message to an assigned performer. You define the message that the notification sends. The message can be an informative note or it can prompt the performer for a response. When a performer responds to a notification activity, the Notification System processes the response and informs the Workflow Engine that the notification activity is complete so that it can continue processing the next eligible activity. See: To Create a Notification Activity, page 3-58.

You specify the performer of a notification activity when you include the notification activity as a node in the process. You can either designate the performer to be a specific role or an item type attribute that dynamically returns the name of a role. See: To Define Nodes, page 4-5 and Roles, page 4-19.

When you define a notification activity, you can also optionally:

- Check Expand Roles to send an individual copy of the notification message to each user in the role. The notification remains in a user's notification queue until the user responds to or closes the notification.

  > **Important:** You should expand roles to send out a broadcast-type message that you want all users of that role to see.
  >
  > If you do not expand the role for a notification activity, Oracle Workflow sends one copy of the notification message to the assigned performer role and that notification is visible in the notification queue of all the users in that role. If one user in that role responds to or closes that notification, the notification is removed from the notification queue of all other users in that role.

- Specify a post-notification function that the Workflow Engine executes in response to an update of the notification's state after the notification is delivered. The Workflow Engine runs the post-notification function in the following modes:
  - `VALIDATE`, `RESPOND`, and `RUN` - When a notification recipient responds to the notification.

    The Workflow Engine initially runs the post-notification function in `VALIDATE` mode which allows you to validate the response values before accepting the response. Then the Workflow Engine runs the post-notification function in `RESPOND` mode to record the response. Finally, when the Notification System completes execution of the post-notification function in `RESPOND` mode, the Workflow Engine then runs the post-notification function again in `RUN` mode.
  - `FORWARD` - When a notification recipient forwards the notification.
  - `TRANSFER` - When a notification recipient transfers the notification.
  - `QUESTION` - When a notification recipient requests more information about the notification from another user.
  - `ANSWER` - When a request recipient responds with more information about the notification.

- `TIMEOUT` - When the notification times out.

For example, if you wish to restrict the roles that a notification can be forwarded to, you can specify a post-notification function that the Workflow Engine executes in `FORWARD` mode when the notification recipient attempts to forward the notification. The post-notification function would audit the role and either allow the forward to occur or reject it with an error. See: Post-Notification Functions, *Oracle Workflow API Reference* and Notification Model, *Oracle Workflow API Reference*.

To create a post-notification function, you should use the same PL/SQL API required for function activities. See: Standard API for PL/SQL Procedures Called by Function Activities, page 6-2.

By both checking Expand Roles and specifying a post-notification function, you can create your own custom vote tallying activity. See: Voting Activity, page 3-69.

Function Activity

A function activity is defined by the PL/SQL stored procedure or external program that it calls. Function activities are typically used to perform fully automated steps in the process. As a PL/SQL stored procedure, a function activity accepts standard arguments and can return a completion result.

If you pass a parameter for the stored procedure, you can expose that parameter as an activity attribute. The activity attribute's value can be set when you define that activity as a node in your process. Note that these activity attributes are available only to the current activity and are not global like item type attributes. See: To Define Activity Attribute Values, page 4-13.

As an external program, a function activity is able to enqueue payload information into an Oracle Advanced Queuing outbound queue for some external agent to dequeue and consume. The external agent can similarly enqueue updated attributes and a completion result into an inbound queue that the Workflow Engine consumes and processes.

As an external Java program, a function activity is able to enqueue payload information into an Oracle Advanced Queuing outbound queue for the Java Function Activity Agent to dequeue and consume. The results of the Java program are enqueued into an inbound queue that the Workflow Engine consumes and processes. This functionality is currently only available for the standalone version of Oracle Workflow. See: To Create a Function Activity, page 3-60.

Event Activity

An event activity represents a business event from the Business Event System within a workflow process. Include event activities in workflow processes to model complex processing or routing logic for business events beyond the standard event subscription actions. See: Managing Business Events, page 8-2.

An event activity can either receive, raise, or send a business event.

Receive Event Activity

A Receive event activity accepts an event sent from the Event Manager. You can send an event to launch one particular new process or continue one particular existing process identified by a specific item type, process name, and item key. You can also send an event to continue one or more existing processes based only on a business key attribute.

A Receive event activity can be marked as a Start activity for a process, meaning it is always enabled to receive events. Alternatively, a Receive event activity can be placed within the process, so that it is only enabled to receive events after the process transitions to that activity.

You can define a Receive event activity to accept only one specific event, to accept only an event that is a member of a specific event group, or to accept any event.

When an event subscription sends an event to a workflow process, the Workflow Engine performs the following processing:

- Sets any parameters in the event message parameter list as item type attributes for the process, creating new item type attributes if a corresponding attribute does not already exist for any parameter.
- Sets the subscription's globally unique identifier (GUID) as a dynamic item attribute so that the workflow process can reference other information in the subscription definition.
- If the event was originally raised by a Raise event activity in another workflow process, the item type and item key for that process are included in the parameter list within the event message. In this case, the Workflow Engine automatically sets the specified process as the parent for the process that receives the event, overriding any existing parent setting. See: SetItemParent, *Oracle Workflow API Reference*.
- Searches for receive event activities that are eligible to accept the event. For an activity to be eligible, the event must match the activity's event filter, and the activity must either be marked as a Start activity or have a status of 'NOTIFIED', meaning the process has transitioned to the event.

  > **Note:** If the event was sent to one or more existing workflow processes based on a business key, rather than to one particular workflow process, then to be eligible an activity must have an event filter that matches the event, a status of 'NOTIFIED' , and a #BUSINESS_KEY attribute that matches the event key. See: Event Subscriptions for Standalone Oracle Workflow, page 8-29 or Event Subscriptions for Oracle Applications, page 8-58.

- If an event arrives at a Start activity to launch a new process instance, the Workflow Engine also searches for all other receive event activities that are marked as Start activities and that do not have any incoming transitions, regardless of their event filter. For these activities, the Workflow Engine sets the activity status to 'NOTIFIED' so that they will be ready to receive an event if any more events are sent to this process. This feature lets you design a workflow process that requires multiple events to be received when you do not know in advance the order in which the events will arrive.
- Stores the event name, event key, and event message in item type attributes, as specified in each eligible activity node's event details.
- Marks all the eligible event activity nodes with a status of 'COMPLETED' and continues the thread of execution from each of those nodes.

If an event activity has already received an event and another matching event is sent to the process, then the On Revisit flag for the activity determines whether the Workflow Engine reexecutes the activity. See: To Define Optional Activity Details, page 3-67.

Example - Using a Business Key

Use a business key to identify which workflow processes should receive an event when the event applies to one or more processes that are already started and are awaiting input to continue. For example, a workflow process initiated for a purchase order with the number PO123 may be awaiting input such as credit card authorization from another workflow or from an external interface, through an event named `po.credit.authorization`. The authorization event should only continue the workflow process instance associated with the PO123 purchase order; it should not continue any other purchase order processes.

To achieve this purpose, the receive event activity in the purchase order workflow process should be defined to accept only the `po.credit.authorization` event and should include an activity attribute named `#BUSINESS_KEY` with a value of `PO123`. The other workflow process or external system that raises the event must set the event key to `PO123`. Additionally, the event subscription to the `po.credit.authorization` event must use the WF_RULE.Instance_Default_Rule rule function or, in Oracle E-Business Suite, the subscription must be defined with the Launch When Business Key Matches option. With this configuration, when the event arrives, it will continue only the workflow process instance associated with the specific purchase order number `PO123` through the `#BUSINESS_KEY` activity attribute.

Raise Event Activity

A Raise event activity retrieves information about the event and raises the event to the Business Event System, which will then execute subscriptions to the event. The activity retrieves the event name, event key, and event data as specified in the node's event details. The event details can be dynamically determined at runtime using item type attributes. You can also specify the event name as a predefined constant for the event activity node.

Additionally, the activity retrieves the names and values of any activity attributes defined for it and sets these attributes as parameters in the parameter list for the event message. If the event message is later received by another process, the Workflow Engine sets the event parameters as item type attributes for that process. See: Event Message Structure, *Oracle Workflow API Reference*.

The activity also automatically sets the item type and item key for the current workflow process in the parameter list for the event message. If the event message is later received by another process, the Workflow Engine uses that item type and item key to automatically set the process that raised the event as the parent for the process that receives the event. See: SetItemParent, *Oracle Workflow API Reference*.

If you want to raise the new event using the event data and parameter list from an existing event message, you can define a special activity attribute named `#EVENTMESSAGE2` for the raise event activity. Set the existing event message as the value of the `#EVENTMESSAGE2` attribute, which must be an attribute of type event. If this attribute is defined, the activity retrieves the event data and parameter list from the specified event and sets them into the new event message before it is raised.

> **Note:** If you also specified event data in the node's event details, however, the activity sets that event data into the event, overriding any event data from the `#EVENTMESSAGE2` attribute. If you specified any additional parameters in activity attributes for the raise event activity, the activity also sets those parameters into the parameter list for the event message, overriding the values of any parameters with the same names from the `#EVENTMESSAGE2` attribute.

For example, if the process previously received an event and you want to raise a new event with that event data, you can define the `#EVENTMESSAGE2` attribute for the raise event activity and set its default value to the item attribute where the received event is stored.

Send Event Activity

A Send event activity retrieves the event name, event key, event message, outbound agent, and inbound agent, as specified in the node's event details. If no correlation ID is initially specified in the event message, the correlation ID is automatically set to the item key of the process. Then the Send event activity sends the event directly from the outbound agent to the inbound agent. The event details can be dynamically determined at runtime using item type attributes. You can also specify the event name, outbound agent, and inbound agent as predefined constants for the event activity node. See: To Create an Event Activity, page 3-62 and To Define Event Details for an Event Node, page 4-9.

> **Note:** A Send event activity does not raise the event to the Business Event System, so no subscription processing is performed.

Process Activity

A process activity represents a collection of activities in a specific relationship. When a process activity is contained in another process it is called a subprocess. In other words, activities in a process can also be processes themselves. There is no restriction on the depth of this hierarchy. See: To Create a Process Activity, page 3-66 and Subprocesses, *Oracle Workflow Administrator's Guide.*

> **Caution:** Oracle Workflow does not support using a subprocess activity multiple times within a process hierarchy.

Activity Cost

Each function activity and event activity has a cost associated with it. The cost is a value representing the number of seconds it takes for the Workflow Engine to execute the activity. If you do not know how long it takes for the Workflow Engine to perform the activity, you can enter an estimated cost and update it later as you accumulate more information about its performance. Generally, you should assign complex, long running activities a high cost.

The valid range for cost is 0.00 to 1,000,000.00.

> **Important:** Although the cost is entered and displayed in seconds in Oracle Workflow Builder, it is actually converted and stored in the database as hundredths of a second.

In normal processing, the Workflow Engine completes the execution of a single activity before continuing to a subsequent activity. In some cases, an activity might take so long to process that background processing would be more appropriate.

You can define your Workflow Engine to defer activities with a cost higher than a designated threshold to a background process. The engine then continues processing the next pending eligible activity that may occur in another parallel branch of the process.

The default threshold for the Workflow Engine is 50 hundredths of a second. Activities with a cost higher than this are deferred to background engines. A background engine

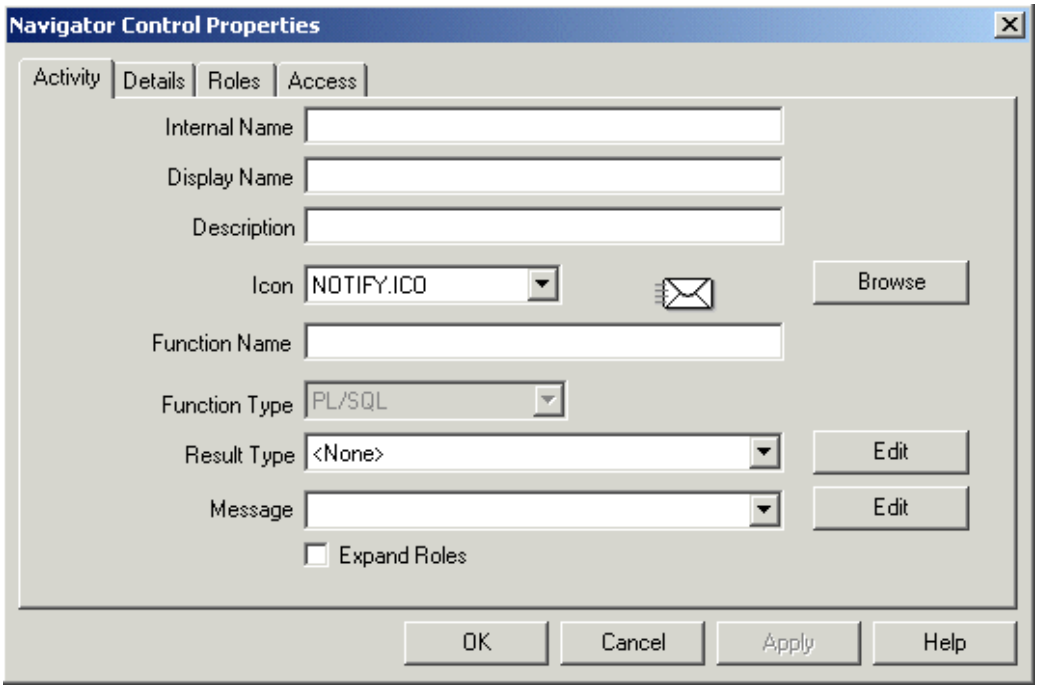
can be customized to execute only certain types of activities. You can set the workflow engine threshold through SQL*Plus. See: Setting Up Background Engines, *Oracle Workflow Administrator's Guide*, To Set Engine Thresholds, *Oracle Workflow Administrator's Guide*, and Deferring Activities, *Oracle Workflow Administrator's Guide*.

Defining Activities

To Create a Notification Activity:

1. Select the item type that you want to create a notification for in the navigator tree, then choose New Notification from the Edit menu. Define your notification activity in the Activity property page that appears.

Notification Activity Property Page



You can also select a message in the navigator tree and drag and drop the message into the Notifications branch of the same item type to create a notification activity that sends that message.

2. A notification activity must have an Internal Name (all uppercase and no leading/trailing spaces) and a Display Name, which is the translatable name that appears in your process diagram. Use the description to provide an explanation about this activity.

> **Important:** To update the internal name for an activity once it is defined, you must use a special SQL script called `wfchact.sql`. You should only use this script to correct errors in an activity's internal name during design time. Do not use this script to rename activities that are involved in running instances of processes. See: Wfchact.sql, *Oracle Workflow Administrator's Guide*.

Caution: Do not include colons ":" or leading/trailing spaces in your internal name.

3. Indicate the result type (a predefined Lookup Type) for this activity. Result types list the possible results returned by this activity. Your workflow diagram may branch depending on the value returned by your completed activity. See: To Create Lookup Types, page 3-20.

   You can choose `<None>` as the result type if your activity does not return a value, or if your workflow process does not depend on the value returned.

4. Select the name of the message you want this notification to send. See: To Create a Message, page 3-42.

5. If you plan to assign this notification to a role consisting of multiple users and you want to send an individual copy of this notification to each user in the role, then check Expand Roles. If you uncheck Expand Roles, then only one copy of the notification is delivered to the role as a whole. See: Notification Activity, page 3-53.

6. You can optionally specify a PL/SQL stored procedure in the Function field. The procedure is known as a post-notification function and lets you couple processing logic to the notification activity. See: Standard API for PL/SQL Procedures Called by Function Activities, page 6-2 and Post-Notification Functions, *Oracle Workflow API Reference*.

   If you check Expand Roles, and you assign a message that has a special Result to this notification activity, then use the Function field to specify the name of a custom PL/SQL stored procedure that tallies the responses you get back from each of the recipients of this notification. Specify the procedure using the format: `<package_name>.<procedure_name>`. See: Voting Activity, page 3-69.

7. Choose an icon that identifies your activity. You can use any icon, as long as the icon is stored in a `.ico` file, to symbolize the action of an activity. See: Adding Custom Icons to Oracle Workflow, *Oracle Workflow Administrator's Guide*.

   Choose Browse to view the icon files listed in the workflow icons subdirectory.

   You can also drag and drop icon files from the Windows Explorer or File Manager onto an activity in your navigator tree to assign that icon to the activity.

8. Choose Apply to save your changes.

9. Select the Details tab to display and modify optional Details of the activity. See: To Define Optional Activity Details, page 3-67.

10. Select the Roles tab page to specify the roles that have access to this notification activity. (This functionality will be supported in a future release.)

11. Select the Access tab page to set the access levels allowed to modify this notification. See: Allowing Access to an Object, page 3-17.

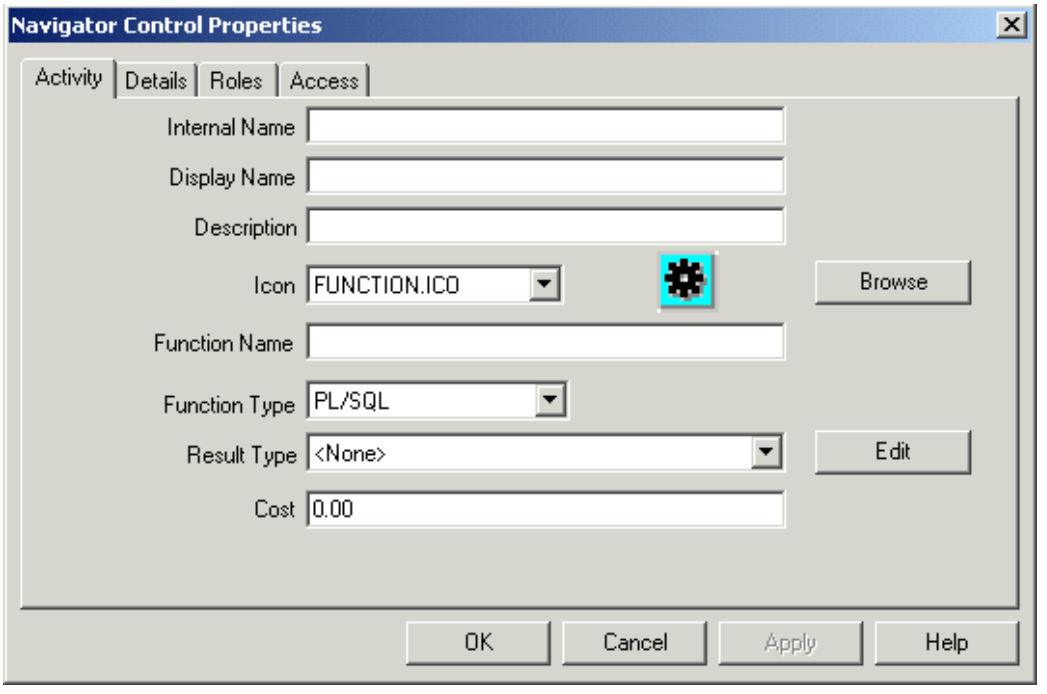
12. Choose OK to save your changes and close the property pages.

13. The notification activity now appears beneath Notifications in the navigator tree. You can review or edit the properties of this activity at any time by double-clicking on the activity in the navigator tree or by selecting the activity and choosing Properties from the Edit menu or by pressing Enter on your keyboard.

To Create a Function Activity:

1. Select the item type that you want to create a function for in the navigator tree, then choose New Function from the Edit menu. Define your function activity in the Activity property page that appears.

Function Activity Property Page



2. A function activity must have an Internal Name (all uppercase and no leading/trailing spaces) and a Display Name, which is the translatable name that appears in your process diagram. Use the description to provide an explanation about this activity.

   Important: To update the internal name for an activity once it is defined, you must use a special SQL script called `wfchact.sql`. You should only use this script to correct errors in an activity's internal name during design time. Do not use this script to rename activities that are involved in running instances of processes. See: Wfchact.sql, *Oracle Workflow Administrator's Guide*.

   Caution: Do not include colons "`:`" or leading/trailing spaces in your internal name.

3. Enter the name of the function you want this activity to execute. In the Type field, specify whether the function is a PL/SQL function, an External function, or an External Java function.

   - For a PL/SQL function, set the function type to `PL/SQL` and specify the function as `<package_name>.<procedure_name>`. The function must be defined according to a standard API. See: Standard API for PL/SQL Procedures Called by Function Activities, page 6-2.

- For an external function activity, set the function type to `External`. The Workflow Engine enqueues an entry in the "Outbound" queue and sets the correlation value of that entry to a value composed of the Workflow schema name and the item type in the following format:

  `<schema_name><item_type>`

  See: Workflow Queue APIs, *Oracle Workflow API Reference*.

  You must create your own queue handler to search for this type of record on the "Outbound" queue. The queue handler must execute the action associated with the record and seed the result of the action onto the "Inbound" queue. The background engine then takes care of messages on the inbound queue and restarts your original workflow process. See: Deferred Processing, *Oracle Workflow API Reference*.

- For an external Java function activity, set the function type to `External Java` and enter the class name of your custom Java class as the function name. This functionality is currently only available for the standalone version of Oracle Workflow. If the custom class is within a package, prefix the class name with the package name in the following format:

  `<customPackage>.<customClass>`

  The Java class must be defined according to a standard API. See: Standard API for Java Procedures Called by Function Activities, page 6-7.

  The Workflow Engine enqueues an entry in the 'Outbound' queue. The Java Function Activity Agent dequeues messages of this type, executes your Java program, and enqueues the results onto the 'Inbound' queue. The background engine then takes care of messages on the inbound queue and restarts your original workflow process. See: Setting Up the Java Function Activity Agent, *Oracle Workflow Administrator's Guide* and Deferred Processing, *Oracle Workflow API Reference*.

  > **Note:** These 'Outbound' and 'Inbound' queues are separate from the queues used for the Business Event System. See: Workflow Queue APIs, *Oracle Workflow API Reference*.

  > **Note:** To execute external Java function activities, you must include your JAR files in your `CLASSPATH`.

4. Indicate the result type (a predefined Lookup Type) for this activity. Result types list the possible results returned by this activity. Your workflow diagram may branch depending on the value returned by your completed activity. See: To Create Lookup Types, page 3-20.

   You can choose `<None>` as the result type if your activity does not return a value, or if your workflow process does not depend on the value returned.

5. Specify the cost of this function activity. See: Activity Cost, page 3-57.

6. Choose an icon that identifies your activity. You can use any icon, as long as the icon is stored in a `.ico` file, to symbolize the action of an activity. See: Adding Custom Icons to Oracle Workflow, *Oracle Workflow Administrator's Guide*.

   Choose Browse to view the icon files listed in the workflow icons subdirectory.

You can also drag and drop icon files from the Windows Explorer or File Manager onto an activity in your navigator tree to assign that icon to the activity.

7. Choose Apply to save your changes.
8. Select the Details tab to display and modify the optional details of the activity. See: To Define Optional Activity Details, page 3-67.
9. Select the Roles tab page to specify the roles that have access to this function activity. (This functionality will be supported in a future release.)
10. Select the Access tab page to set the access levels allowed to modify this function. See: Allowing Access to an Object, page 3-17.
11. The function activity now appears beneath Functions in the navigator tree. You can review or edit the properties of this activity at any time by double-clicking on the activity in the navigator tree or by selecting the activity and choosing Properties from the Edit menu or by pressing Enter on your keyboard.
12. If your function requires input arguments, you can expose those arguments in Oracle Workflow Builder as attributes of the function activity. Function activity attributes behave as parameters whose values you can modify for each usage of the activity in a process. Function activity attributes are specific to a function activity and are not global to a process. See: To Define an Item Type or Activity Attribute, page 3-7.

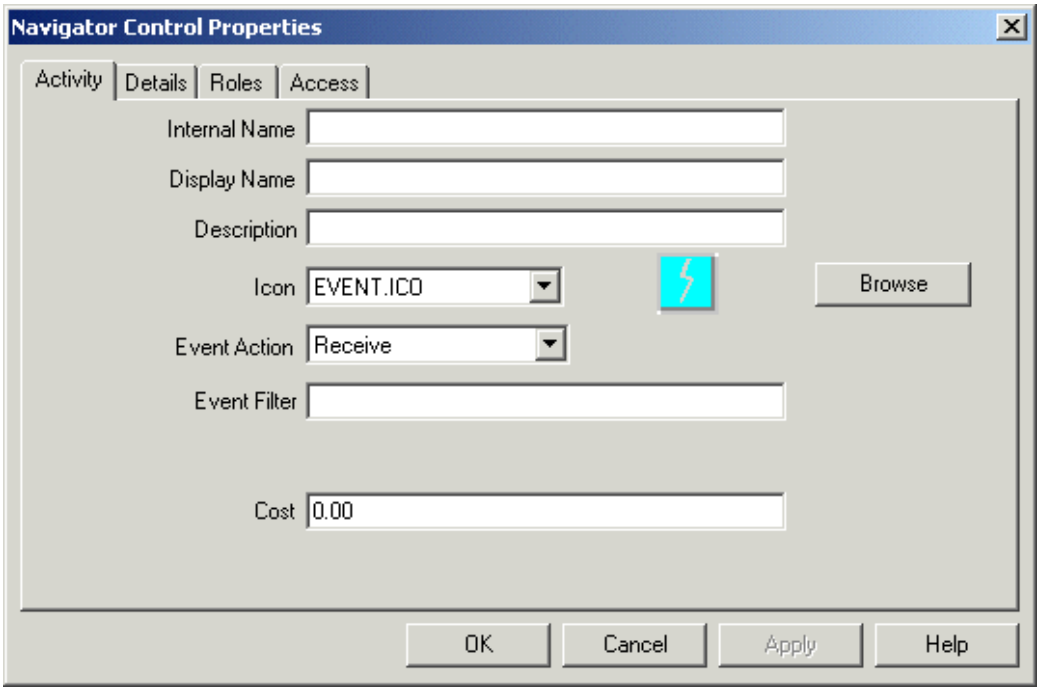
    To create a function activity attribute that references an item type attribute, select the referenced item type attribute in the navigator tree, and hold down your mouse select button as you drag the item type attribute to your function activity. The Default Value region is automatically set to Item Attribute and references the originating item attribute.

    When you include a function activity as a node in a process, you can assign a value to the function activity attribute that is specific to that node. See: To Define Activity Attribute Values, page 4-13.

To Create an Event Activity:

1. Select the item type that you want to create an event for in the navigator tree, then choose New Event from the Edit menu. Define your event activity in the Activity property page that appears.

Event Activity Property Page



2. An event activity must have an Internal Name (all uppercase and no leading/trailing spaces) and a Display Name, which is the translatable name that appears in your process diagram. Use the description to provide an explanation about this activity.

   Important: To update the internal name for an activity once it is defined, you must use a special SQL script called `wfchact.sql`. You should only use this script to correct errors in an activity's internal name during design time. Do not use this script to rename activities that are involved in running instances of processes. See: Wfchact.sql, *Oracle Workflow Administrator's Guide*.

   Caution: Do not include colons "`:`" or leading/trailing spaces in your internal name.

3. Choose an icon that identifies your activity. You can use any icon, as long as the icon is stored in a `.ico` file, to symbolize the action of an activity. See: Adding Custom Icons to Oracle Workflow, *Oracle Workflow Administrator's Guide*.

   Choose Browse to view the icon files listed in the workflow icons subdirectory.

   You can also drag and drop icon files from the Windows Explorer or File Manager onto an activity in your navigator tree to assign that icon to the activity.

4. Select the Event Action for the activity.

   - Receive - Receive an event from the Business Event System.
   - Raise - Raise an event to the Business Event System.
   - Send - Send an event directly from one Event agent to another agent without re-raising the event to the Business Event System.

Note: Depending on the event action you select, you may need to define item type attributes for some or all of the following event details:

- Event Name
- Event Key
- Event Message
- Event Data
- Out Agent
- To Agent

When you include the event activity as a node in a process, you can use the item type attributes to specify where to store or retrieve the required event detail information for that node. The item type attributes that you use for event details must be associated with the same item type as the event activity itself. See: To Define an Item Type or Activity Attribute, page 3-7 and To Define Event Details for an Event Node, page 4-9.

5. If you are defining a Receive event activity, you can optionally enter an Event Filter to specify the event that the activity can receive.
   - To allow the activity to accept only one specific event, enter the full internal event name.
   - To allow the activity to accept any event that is a member of a specific event group. enter the full internal event group name.
   - To allow the activity to accept any event, leave the Event Filter field blank.

   See: To Define an Event, page 8-5.
6. Enter an optional cost for the activity. For event activities with the event actions Raise or Send, you can use the cost to defer long running activities to a background engine. See: Activity Cost, page 3-57.
7. Choose Apply to save your changes.
8. Select the Details tab to display and modify the optional details of the activity. See: To Define Optional Activity Details, page 3-67.
9. Select the Roles tab page to specify the roles that have access to this function activity. (This functionality will be supported in a future release.)
10. Select the Access tab page to set the access levels allowed to modify this event. See: Allowing Access to an Object, page 3-17.
11. The event activity now appears beneath Events in the navigator tree. You can review or edit the properties of this activity at any time by double-clicking on the activity in the navigator tree or by selecting the activity and choosing Properties from the Edit menu or by pressing Enter on your keyboard.
12. For a raise event activity, if the event raised by the activity requires additional parameters to be included in the event message, you can define those parameters as attributes of the raise event activity. When the event is raised, the activity attributes are set as parameters in the parameter list for the event message. If the event

message is later received by another process, the Workflow Engine sets the event parameters as item type attributes for that process. You can modify the values of the attributes for each usage of the raise event activity in a process. Event activity attributes are specific to an event activity and are not global to a process. See: To Define an Item Type or Activity Attribute, page 3-7.

> **Note:** A Raise event activity also automatically sets the item type and item key for the current workflow process in the parameter list for the event message. If the event message is later received by another process, the Workflow Engine uses that item type and item key to automatically set the process that raised the event as the parent for the process that receives the event. See: SetItemParent, *Oracle Workflow API Reference*.

If you want to raise the new event using the event data and parameter list from an existing event message, you can also define a special activity attribute named `#EVENTMESSAGE2` for the raise event activity. Set the existing event message as the value of the `#EVENTMESSAGE2` attribute, which must be an attribute of type event. If this attribute is defined, the activity retrieves the event data and parameter list from the specified event and sets them into the new event message before it is raised.

> **Note:** If you also specified event data in the node's event details, however, the activity sets that event data into the event, overriding any event data from the `#EVENTMESSAGE2` attribute. If you specified any additional parameters in activity attributes for the raise event activity, the activity also sets those parameters into the parameter list for the event message, overriding the values of any parameters with the same names from the `#EVENTMESSAGE2` attribute.

To create an event activity attribute that references an item type attribute, select the referenced item type attribute in the navigator tree, and hold down your mouse select button as you drag the item type attribute to your event activity. The Default Value region is automatically set to Item Attribute and references the originating item attribute.

When you include an event activity as a node in a process, you can assign a value to the event activity attribute that is specific to that node. See: To Define Activity Attribute Values, page 4-13.

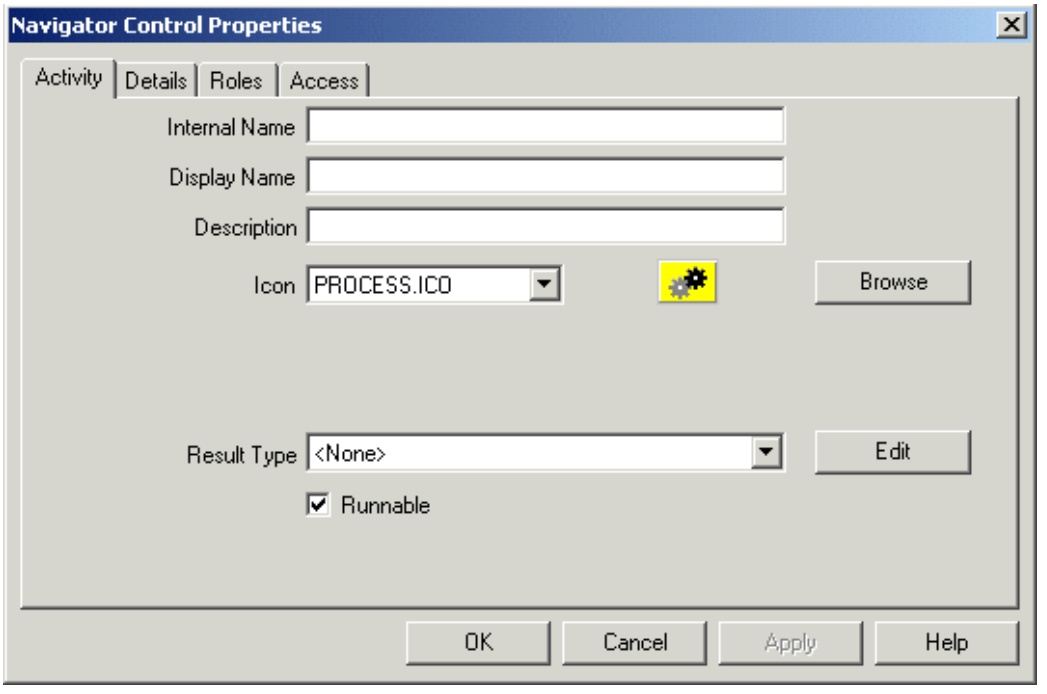
13. For a receive event activity, if you want to match the event with one or more workflow processes based on a business key rather than sending the event to one particular workflow process, define a special activity attribute named `#BUSINESS_KEY`. This attribute must be of type text. Set the default value of the activity attribute to an item type attribute, and include logic in your workflow process to set that item type attribute to an appropriate business key value at runtime. For the workflow process to receive an event, this business key must match the event key. See: To Define an Item Type or Activity Attribute, page 3-7 and Event Subscriptions for Standalone Oracle Workflow, page 8-29 or Event Subscriptions for Oracle Applications, page 8-58.

To Create a Process Activity:

Before you can draw a workflow process diagram, you must first create a process activity in the navigator tree to represent the process diagram.

1. Select the item type that you want to create a process activity for in the navigator tree, then choose New Process from the Edit menu. Define your process activity in the Activity property page that appears.

Process Activity Property Page



If a process activity is closed and you want to redisplay it, select the process activity in the navigator tree and press Enter or select Properties from the mouse menu button.

2. A process activity must have an Internal Name (all uppercase and no leading/trailing spaces) and a Display Name, which is the translatable name that appears in your process diagram. Use the description to provide an explanation about this activity.

   Important: To update the internal name of an activity once it is defined, you must use a special SQL script called `wfchact.sql`. You should only use this script to correct errors in an activity's internal name during design time. Do not use this script to rename activities that are involved in running instances of processes. See: Wfchact.sql, *Oracle Workflow Administrator's Guide*.

   Caution: Do not include colons "`:`" or leading/trailing spaces in your internal name.

3. Indicate the result type (a predefined Lookup Type) for this activity. Result types list the possible results returned by this process. See: To Create Lookup Types, page 3-20.

You can choose `<None>` as the result type if you do not need to record any specific result for the completion of your process.

4. Choose an icon that identifies your activity. You can use any icon, as long as the icon is stored in a `.ico` file, to symbolize the action of an activity. See: Adding Custom Icons to Oracle Workflow, *Oracle Workflow Administrator's Guide*.

   Choose Browse to view the icon files listed in the workflow icons subdirectory.

   You can also drag and drop icon files from the Windows Explorer or File Manager onto an activity in your navigator tree to assign that icon to the activity.

5. Check Runnable so that the process that this activity represents can be initiated as a top-level process and run independently. If your process activity represents a subprocess that should only be executed if it is called from a higher level process, then uncheck Runnable. See: CreateProcess, *Oracle Workflow API Reference*.

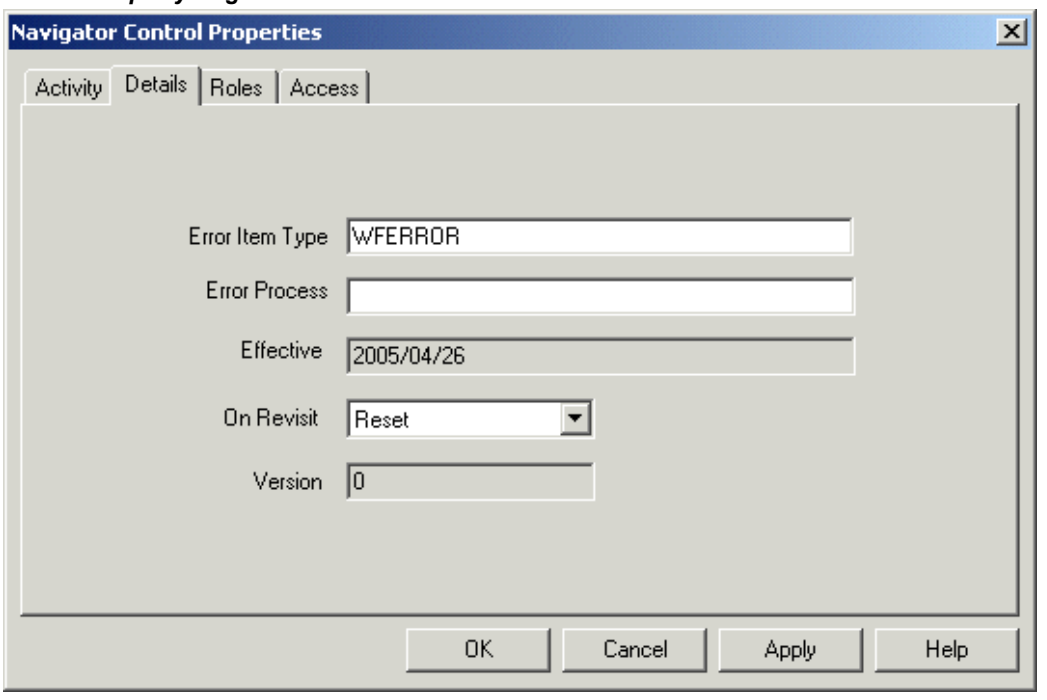
   > **Caution:** Oracle Workflow does not support reusing a subprocess activity multiple times within a process hierarchy. If you wish to use a subprocess more than once in a process, you must create a distinct copy of the subprocess for each instance needed.

6. Choose Apply to save your changes.
7. Select the Details tab to display and modify the optional details of the activity. See: To Define Optional Activity Details, page 3-67.
8. Select the Access tab page to set the access levels allowed to modify this process. The access you set for a process activity determines who has access to edit its process diagram. See: Allowing Access to an Object, page 3-17.
9. Choose Apply to save your changes, OK to save your changes and close the property page, or Cancel to cancel your changes and close the property page.
10. The process activity now appears beneath Processes in the navigator tree. You can review or edit the properties of this activity at any time by selecting the activity and choosing Properties from the Edit menu or by pressing Enter on your keyboard.

To Define Optional Activity Details:

1. Select the Details tab in the activity's property pages.

Details Property Page



2. If you are creating a process activity, you can specify an error process to execute in the event that an error occurs in the current process. Enter the internal name of the item type that owns the error process and then specify the internal name of the error process activity to execute. Note that the error process item type does not need to be open in your current Oracle Workflow Builder session for you to define it here. See: Error Handling for Workflow Processes, page 11-1.

   Note: Both the error item type and the error process must be specified for an error process to be launched when an error occurs. The error item type defaults to `WFERROR`, the internal name for the System: Error item type. If you want to launch one of the predefined error processes provided in this item type, you must enter the internal name of that process. You can also enter a custom error item type and process.

   If you assign one of the predefined error processes to your activity, you can customize the behavior of the error process by defining two special item type attributes within your own item type. A `WF_ADMINISTRATOR` attribute lets you specify the role to which Oracle Workflow sends an error notification, and an `ERROR_TIMEOUT` attribute lets you specify whether the error notification times out. See: Customizing Error Notifications for an Item Type, page 11-3.

3. The effective date tells you when this version of the activity is available for the Workflow Engine to execute. If the Effective Date field is blank, the activity is effective immediately.

You set the effective date when you save your changes using the Save As option in the File menu. All your activity modifications share the same effective date when you save.

4. Select a value for On Revisit to determine how the Workflow Engine handles this activity when it is transitioned to more than once. If this activity is the first activity that is revisited, as in a loop, you should set On Revisit to specify how you want the Workflow Engine to process the loop. The first activity in a loop is also called the pivot activity. For all other activities in a loop, the value of On Revisit is irrelevant.
   - If On Revisit is set to `Ignore`, the Workflow Engine executes the activity only once, and ignores the activity for all subsequent revisits.
   - If On Revisit is set to `Reset`, the Workflow Engine resets the completed activities in the loop by traversing through the loop in forward order from the pivot activity, executing each activity in `CANCEL` mode. You can include special logic in each function's `CANCEL` mode to undo prior operations. The Workflow Engine then traverses through the loop in forward order, reexecuting each activity, starting with the pivot activity, in `RUN` mode.
   - If On Revisit is set to `Loop`, the Workflow Engine simply reexecutes the pivot activity and all activities that follow in the loop, without resetting, as if they have never been executed before. See: Looping, *Oracle Workflow API Reference*.
5. The version number identifies which revision of the activity you are examining. The engine ensures that it uses the most recent updates to an activity by using the latest effective version number of that activity.
6. Choose Apply to save your changes.

To Copy an Activity:

1. Select the activity to copy in the navigator tree.
2. Hold down your mouse select button as you drag the activity to the item type branch you want to copy it to.
3. If you copy the activity within the same item type, a property page will appear prompting you for a new unique internal and display name for the copied activity.

   Note: You can also use the Copy and Paste options in the Edit menu.

4. When you are done, choose OK.

   Note: Copying a function, event, or notification activity also copies any attributes or message associated with it, respectively.

Related Topics

Using the Edit Button in a Property Page, page 2-6

Voting Activity

You can create a voting activity that lets you send a notification to a group of users in a role and tally the responses from those users. The results of the tally determine the activity that the process transitions to next.

A voting activity is a notification activity that first sends a notification message to a group of users and then performs a PL/SQL post-notification function to tally the users' responses (votes).

The activity attributes you define and the following four fields in the property pages of the notification activity determine its voting behavior:

- Message field
- Result Type field
- Expand Roles check box
- Function field

Creating a Voting Activity:

1. Create a voting lookup type that contains the responses you want to tally in your voting activity. See: To Create Lookup Types, page 3-20.
2. Create a voting message that prompts a recipient to respond with one of the values in the voting lookup type. Complete the Result tab for the message. Set the lookup type in the Result tab to the voting lookup type defined in Step 1. See: To Create a Message, page 3-42.
3. Select the item type that you want to create a voting activity for in the navigator tree, then choose New Notification from the Edit menu.
4. Specify an Internal Name (all uppercase and no leading/trailing spaces) and a Display Name. Use the description to provide an explanation about this voting activity.

   Important: To update the internal name for an activity once it is defined, you must use a special SQL script called `wfchact.sql`. You should only use this script to correct errors in an activity's internal name during design time. Do not use this script to rename activities that are involved in running instances of processes. See: Wfchact.sql, *Oracle Workflow Administrator's Guide*.

   Caution: Do not include colons "`:`" or leading/trailing spaces in your internal name.

5. The Result Type field must contain the lookup type that lists the responses that you want the voting activity to tally. This is the voting lookup type defined in Step 1.
6. Choose an icon that identifies your voting activity.
7. In the Message field, select the name of the voting message you created in Step 2. The voting message prompts the recipient for a response. The response choices are one of the predefined values specified in your voting lookup type.
8. Check Expand Roles so that the Workflow Engine polls for responses from the multiple users in the role rather than just from the first user in the role that replies. See: Notification Activity, page 3-53.
9. In the Function field, specify a function that tallies the responses from users. You can use the PL/SQL procedure *WF_STANDARD.VOTEFORRESULTTYPE* that the Standard Vote Yes/No activity calls. *WF_STANDARD.VOTEFORRESULTTYPE* is a generic tallying function. The Result Type that you specify for the voting activity

defines the possible responses for the function to tally. The activity attributes that you define for the voting activity determine how the function tallies the responses. See: Vote Yes/No Activity, page 5-8.

Alternatively, you can specify your own custom tallying function, but you should make sure it conforms to the standard API for function activities. Specify the procedure using the format: `<package_name>.<procedure_name>`. See: Standard API for PL/SQL Procedures Called by Function Activities, page 6-2.

10. Choose Apply to save your changes.
11. Select the Details tab to display and modify the Details property page of the activity. See: To Define Optional Activity Details, page 3-67.
12. Select the Roles tab page to specify the roles that have access to this notification activity. (This functionality will be supported in a future release.)
13. Select the Access tab page to set the access levels allowed to modify this notification. See: Allowing Access to an Object, page 3-17.
14. If you use the *WF_STANDARD.VOTEFORRESULTTYPE* tallying function, create a custom activity attribute of type Number for each possible voting response. Remember that each possible voting response is a lookup code associated with the voting activity's result type. Hence, when you define your custom activity attribute, the internal name of the activity attribute must match the internal name of the lookup code, that is, the response value.

    The value of the activity attribute can either be blank or a number that represents the percentage of votes required for a particular result. If you provide a percentage, then the result is matched if the actual tallied percentage for that response is greater than your specified percentage. If you leave an activity attribute value blank, then the Workflow Engine treats the response for that activity attribute as a default. In other words, if no particular percentage is satisfied after the votes are tallied, then the response that received the highest number of votes among those associated with a blank activity attribute becomes the result.

    > **Note:** If the tallied votes do not satisfy any response percentages and there are no default responses (blank activity attributes) specified, the result is `#NOMATCH`. If a `<No Match>` transition from the voting activity exists, then the Workflow Engine takes this transition, otherwise, it takes the `<Default>` transition. If no `<Default>` transition exists, it raises an error that no transition for the result is available (`ERROR:#NOTRANSITION`).

    > **Note:** If the tallied votes satisfy more than one response percentage or if no response percentage is satisfied, but a tie occurs among the default responses, the result is `#TIE`. If a `<Tie>` transition from the voting activity exists, then the Workflow Engine takes this transition, otherwise, it takes the `<Default>` transition. If no `<Default>` transition exists, it raises an error that no transition for the result is available (`ERROR:#NOTRANSITION`).

15. If you use the *WF_STANDARD.VOTEFORRESULTTYPE* tallying function, then in addition to defining your set of custom activity attributes, you must also define an activity attribute called Voting Option, whose internal name must be

VOTING_OPTION. You can also copy the Voting Option activity attribute from the Vote Yes/No standard activity.

The Voting Option activity attribute specifies how the votes are tallied. The possible values are:

- "Wait for All Votes" - the Workflow Engine waits until all votes are cast before tallying the results as a percentage of all the users notified. If a timeout condition occurs, the Workflow Engine calculates the resulting votes as a percentage of the total votes cast before the timeout occurred.
- "Tally on Every Vote" - the Workflow Engine keeps a running tally of the cumulative responses as a percentage of all the users notified. If a timeout condition occurs, then the responses are tallied as a percentage of the total number of votes cast. Note that this option is meaningless if any of the custom response activity attributes have a blank value.
- "Require All Votes" - the Workflow Engine evaluates the responses as a percentage of all users notified only after all votes are cast. If a timeout condition occurs, the Workflow Engine progresses along the standard timeout transition, or if none is available, raises an error, and does not tally any votes.

Related Topics

Example Voting Methods, page 3-72

Vote Yes/No Activity, page 5-8

Example Voting Methods

1. **Simple Majority**

   The following table shows the custom response activity attribute value assigned to each response for a simple majority voting method.

   Simple Majority Voting Method

Response	Custom Response Activity Attribute Value
A	50
B	50
C	50

   The result is any response that gets more than fifty percent of the votes. If no response gets more than fifty percent, the result is that no match is found (#NOMATCH).

2. **Simple Majority with Default**

   The following table shows the custom response activity attribute value assigned to each response for a simple majority with default voting method.

Simple Majority with Default Voting Method

Response	Custom Response Activity Attribute Value
A	50
B	50
C	blank

If response A gets more than fifty percent of the votes, A is the result. Similarly if response B gets more than fifty percent of the votes, B is the result. If neither response A nor B gets more than fifty percent of the votes, then C is the result.

3. **Simple Majority with Multiple Defaults**

   The following table shows the custom response activity attribute value assigned to each response for a simple majority with multiple defaults voting method.

Simple Majority with Multiple Defaults Voting Method

Response	Custom Response Activity Attribute Value
A	50
B	blank
C	blank

If response A gets more than fifty percent of the votes, A is the result. If A gets fifty percent of the votes, or less, then response B or C is the result depending on which of the two received the higher number of votes. If A gets fifty percent of the votes, or less, and both B and C receive the same number of votes, then the result is a tie (`#TIE`).

4. **Popularity**

   The following table shows the custom response activity attribute value assigned to each response for a popularity voting method.

Popularity Voting Method

Response	Custom Response Activity Attribute Value
A	blank
B	blank
C	blank

The result is the response that gets the highest number of votes.

5. **Black Ball**

   The following table shows the custom response activity attribute value assigned to each response for a black ball voting method.

Black Ball Voting Method

Response	Custom Response Activity Attribute Value
YES	100
NO	0

Any vote for response `NO` makes `NO` the result.

6. **Jury**

   The following table shows the custom response activity attribute value assigned to each response for a jury voting method.

Jury Voting Method

Response	Custom Response Activity Attribute Value
GUILTY	100
NOT_GUILTY	100

A unanimous response is required, otherwise no match is found (`#NOMATCH`).

Related Topics

Voting Activity, page 3-69

Vote Yes/No Activity, page 5-8

Deleting Objects in Oracle Workflow Builder

You can delete an object in Oracle Workflow Builder even if the object is referenced by other objects, assuming the object is not protected against customizations. If the object you want to delete is referenced by other objects, a Workflow Error dialog box appears, warning you about the foreign key references that will break. You can proceed to delete the object anyway or cancel the action. If you choose to delete, then when you save or verify the workflow process definition, a Workflow Error dialog box appears, reporting all broken foreign key references that exist in the definition.

As a result of this behavior, you can load workflow definitions with invalid foreign keys into Oracle Workflow Builder to correct. Oracle Workflow Builder preserves the original internal name reference for any missing foreign key, and displays it in a validation error message when you load the process definition. You can restore a broken foreign key reference in a process definition by recreating the deleted object with its original internal name under its original item type.

You can also delete an entire item type definition in Oracle Workflow Builder.

> **Note:** If you want to delete an item type attribute from a workflow definition in a database, you must use Oracle Workflow Builder to connect to that database in order to perform the deletion. Deleting an item attribute from a workflow definition stored in a flat file and then uploading that flat file definition to a database will not delete the item attribute from the definition stored in the database. To delete an item

attribute completely, you must delete it from your flat file definition and also delete it specifically from any databases in which that workflow item type is loaded while connected to those databases.

Modifying Objects in Oracle Workflow Builder

Before you modify the definitions of any Workflow objects, you should ensure that your changes will not adversely affect any active work items that are based on those definitions. Changes to Oracle Workflow objects have different effects on active work items depending on whether or not the objects support versioning.

For a Workflow object, versioning means that either the object itself or the object that owns it supports multiple occurrences of the same object in the database, distinguished only by a version number, begin date, and end date. For example, the following table shows two versions of a Vote activity that could exist simultaneously in the WF_ACTIVITIES table.

Sample Activity Versions

Name	Version	Begin Date	End Date	Message	Lookup Type
Vote	1	01-JAN-1998	31-DEC-1998	Vote Message	Yes/No
Vote	2	01-JAN-1999	*<blank>*	New Vote Message	Approval

When you modify a Workflow object that supports versioning, both the original version and the new version exist in the database. Any active work items that reference that object will continue to completion still using the same version that was in effect when the work items were initiated. Only new work items initiated after the change will use the new version.

In the above example, work items that are initiated between January 1, 1998 and December 31, 1998 will send the message *Vote Message* with result options of *Yes* or *No*, whether the work items are completed before January 1, 1999 or not. Only work items that are initiated on or after January 1, 1999 will send the message *New Vote Message* with result options of *Approve* or *Reject*.

> **Note:** All process definition information is versioned.

When you modify a Workflow object that does not support versioning, however, the previous definition of the object is updated and only the modified definition exists in the database. Any active work items that reference that object will use the modified object after the change.

If the modified object is no longer compatible with the rest of the workflow definition used by the work item, errors may arise. To avoid such errors, you must take all references to the object into consideration when planning your changes to ensure that the changes do not cause any incompatibility.

> **Note:** If your situation allows, you can avoid the risk of backward incompatibility by aborting and restarting all active work items after you make your changes. This method forces the restarted work items to use the modified definitions of all Workflow objects, including those that support versioning as well as those that do not.

Workflow Objects That Support Versioning

The following Workflow objects support versioning:

- Notifications
- Functions
- Events
- Processes and subprocesses
- Process activities (nodes)
- Activity attributes
- Activity attribute values
- Activity transitions

When you modify any of these objects in the Workflow Builder and save them to the database, the Workflow Loader does not update the existing definition of the object. Instead, a new version of the object or owning object is created.

As a result, you can modify any of these objects without affecting active work items that were initiated before the change.

For example:

- If you update a notification activity to reference a new message, the notification will be versioned.
- If you update a function activity to reference a new lookup type, the function will be versioned.
- If you update a function activity to reference a new API, the function will be versioned.
- If you remove a process activity, or node, from a process diagram, the owning process will be versioned, as well as all the process activities that exist within the process.
- If you add an activity attribute to an activity, the owning activity will be versioned.

The modifications in all of these examples will affect only work items that are initiated after the changes are made.

Workflow Objects That Do Not Support Versioning

The following Workflow objects do not support versioning:

- Item attributes
- Messages
- Lookups
- PL/SQL code referenced by function activities

When you modify any item attributes, messages, or lookups in the Workflow Builder and save them to the database, the Workflow Loader updates the existing definition of the object. Also, if you modify the existing PL/SQL API for a function activity, the function activity will reference the updated API stored in the database.

As a result, if you modify any of these objects, your changes immediately affect any active work items that reference the object. Plan your changes carefully to ensure that the changes do not cause any backward incompatibility.

> **Note:** The Workflow Builder does not support the editing of PL/SQL code. PL/SQL code is listed as a Workflow object here solely for the purpose of explaining the consequences of changing the code referenced by a Workflow function activity.

Item Attributes

When a work item is initiated, Oracle Workflow creates a runtime copy of each item attribute that is defined for that item type. The Workflow Engine refers to these runtime copies whenever an item attribute is referenced in the PL/SQL code for a function activity in the workflow process.

Adding a new item attribute after work items have been initiated will not affect the active work items. However, these work items will not include the new item attribute unless you specifically create the attribute for them by calling the *AddItemAttr()* or *AddItemAttributeArray* APIs. If you also add references to the new item attribute in the existing PL/SQL code within the workflow process, those references may cause errors in active work items that do not include the attribute.

For example, if you change the PL/SQL API for a function activity by calling a Workflow Engine API to communicate with a new item attribute, the function activity will fail for active work items that do not have the new item attribute defined.

You should plan carefully when making any modifications to the existing PL/SQL code within a workflow process to ensure that you do not add references to a new item attribute without also creating the item attribute for active work items that require it. See: PL/SQL Code, page 3-79.

> **Note:** You can, however, add references to new item attributes in the API that starts a workflow process, without making special provisions for active work items. For example, you can call the *SetItemAttribute* or *SetItemAttributeArray* APIs to populate the new item attributes at the start of the process. Active work items will not be affected by such changes, since these work items have already passed through this code.

Messages

When the Workflow Engine requests the Notification System to send a message, the Notification System creates a notification attribute in the notification tables for every message attribute. The notification attribute rows contain the variable data that will be token-replaced into the message body, including the subject line and body text, at runtime.

The message body, however, is not copied into the notification tables. Instead, the message body is referenced by the various Notification System APIs at runtime, when the notification is displayed to the user. As a result, any modifications to a message body will affect notifications in active work items that were sent before the change, as well as notifications that are sent after the change.

You can make certain types of modifications to a message body without risking incompatibility errors. These modifications include:

- Adding static text

- Editing static text
- Removing static text
- Removing message attribute tokens

For example, if you add a static sentence such as "Please approve within five days" to a message body, all notifications in active work items will include the additional sentence when you access the notifications. The Notification System can display the modified message body without any errors because no further information is required to resolve the additional sentence.

However, inappropriate modifications, such as adding tokens for newly created message attributes, may cause notifications in active work items to be displayed incorrectly. You should plan your changes carefully to avoid errors.

If you need to add tokens for new message attributes to a message body, you should implement the change by creating a new message rather than by modifying the existing message. You can attach the new message to your existing notification activity without affecting active work items, since notification activities support versioning.

For example, if you create a new message attribute such as *Approver Name* and you add a token for that attribute in the message body, all notifications in active work items will include the new token when you access the notifications. However, notifications that were sent before the change will not include the new message attribute *Approver Name* as a notification attribute. The Notification System will not be able to resolve the reference to the new message attribute and will display the token "`&APPROVER_NAME`" in the notifications instead.

In this example, instead of modifying the original message body, you should create a new message that includes the new message attribute, add the token for the new attribute to the body of the new message, and attach the new message to your notification activity. When you save your changes, the notification activity will be versioned. Then active work items will continue to reference the old version of the notification activity, and the incompatible token will not appear in those notifications.

Lookup Types and Codes

Lookup types have the following important uses in Oracle Workflow:

- Determining the possible completion statuses (lookup codes) for workflow activities
- Determining the possible completion statuses (lookup codes) for 'Respond' message attributes.

Inappropriate modifications to lookup types may cause active work items that reference those lookup types to fail.

To avoid errors caused by backward incompatibility, follow these guidelines for lookup types that are referenced by active work items:

- Do not delete lookup types.
- Do not delete lookup codes from existing lookup types.
- Do not add lookup codes to existing lookup types.

If you need to make any of the above changes, you should implement the change by creating a new lookup type rather than by modifying the existing lookup type.

You can attach new lookup types to existing activities without affecting active work items, since activities support versioning. However, you should not attach new lookup types to existing message attributes, since Workflow messages do not support versioning.

The following examples show some errors that can be caused by inappropriate lookup type modifications:

- If you add a lookup code to a lookup type that is referenced by a 'Respond' message attribute, the new lookup code will be available for users to select as a response to a notification. However, previous versions of the notification activity will not have branching logic modeled for the new lookup code. If a user selects the new code, the process will enter an '`ERROR`' state.
- If you delete a lookup code from a lookup type that is referenced by a 'Respond' message attribute, users will no longer be able to choose that result code to respond to a notification.

PL/SQL Code

Although function activities support versioning, the underlying PL/SQL code does not support versioning, unless you implement versioning for your code yourself. Modifying PL/SQL code without versioning changes the business flow for active work items that reference that code. Inappropriate modifications may cause these work items to fail.

To prevent changes in the PL/SQL API for a function activity from affecting active work items, you should implement the changes by creating a new API rather than by modifying the existing API. You can attach the new API to your existing function activity without affecting active work items, since function activities support versioning.

If you need to modify an existing API and you cannot create a new API instead, you should plan your changes carefully to ensure that the changes do not cause any backward incompatibility.

For example, if you plan to add a lookup code to the group of values that an API can return, you should first ensure that the function activity node has an outgoing transition, such as 'Default,' that the Workflow Engine can follow when the API returns that value. Otherwise, the process will enter an '`ERROR`' state when that value is returned. If there is no appropriate outgoing transition, you must implement the change in a new API and update the process to model branching logic for the additional return value.

Also, if you change the existing PL/SQL API for a function activity by calling a Workflow Engine API to communicate with a new item attribute, you should ensure that you also create the new item attribute for active work items. Otherwise, the function activity will fail for active work items which do not have the new item attribute defined.

Calls to any of the following APIs with newly created item attributes as parameters may cause the function activity to fail if active work items do not have the item attributes defined:

- wf_engine.SetItemAttrText
- wf_engine.SetItemAttrNumber
- wf_engine.SetItemAttrDate
- wf_engine.SetItemAttrEvent
- wf_engine.SetItemAttrFormattedDate

- wf_engine.SetItemAttrDocument
- wf_engine.SetItemAttrTextArray
- wf_engine.SetItemAttrNumberArray
- wf_engine.SetItemAttrDateArray
- wf_engine.GetItemAttrText
- wf_engine.GetItemAttrNumber
- wf_engine.GetItemAttrDate
- wf_engine.GetItemAttrEvent
- wf_engine.GetItemAttrDocument
- wf_engine.GetItemAttrClob
- wf_engine.GetItemAttrInfo

To create copies of a new item attribute for active work items, call *AddItemAttr()* or one of the *AddItemAttributeArray* APIs. You can place this call either in a custom upgrade script or in an exception handler.

- **Upgrade script** - Before you modify your API, write and execute a custom upgrade script that creates and populates the new item attribute for any active work items that reference that API. The following example shows one way to structure an upgrade script.

```
for <each active work item>
  begin
    wf_engine.AddItemAttr(itemtype,
                          itemkey,
                          '<new_attribute_name>');
    wf_engine.SetItemAttrText(itemtype,
                              itemkey,
                              '<new_attribute_name>',
                              '<New attribute value>');
  end;
end loop;
```

> **Note:** Active work items are identified as those items for which `WF_ITEMS.END_DATE` is null.

- **Exception handler** - After the reference to the new item attribute in your modified API, add an exception handler to create and populate the attribute when the attribute does not already exist. The following example shows one way to structure such an exception handler.

```
procedure <procedure_name>
  (itemtype in varchar2,
   itemkey in varchar2,
   actid in number,
   funcmode in varchar2,
   result in out varchar2)
is

begin
```

```
  --
  -- RUN mode - normal process execution
  --
  if (funcmode = 'RUN') then
    -- your run code goes here
    null;
      wf_engine.SetItemAttrText(itemtype,
        itemkey,
        '<existing_attribute_name>',
        '<Existing attribute value>');
      begin
        wf_engine.SetItemAttrText(itemtype,
          itemkey,
          '<new_attribute_name>',
          '<New attribute value>');
      exception
      when others then
        if (wf_core.error_name = 'WFENG_ITEM_ATTR') then
           wf_engine.AddItemAttr(itemtype,
             itemkey,
             '<new_attribute_name>');
           wf_engine.setitemattrtext(itemtype,
             itemkey,
             '<new_attribute_name>',
             '<New attribute value>');
        else
          raise;
        end if;
      end;
    -- example completion
    result := 'COMPLETE:';
    return;
  end if;

  --
  -- CANCEL mode - activity 'compensation'
  --
  -- This is in the event that the activity must be undone,
  -- for example when a process is reset to an earlier point
  -- due to a loop back.
  --
  if (funcmode = 'CANCEL') then
    -- your cancel code goes here
    null;
    -- no result needed
    result := 'COMPLETE';
    return;
  end if;

  --
  -- Other execution modes may be created in the future.  Your
  -- activity will indicate that it does not implement a mode
  -- by returning null
  --
  result := '';
  return;

exception
```

```
  when others then
    -- The line below records this function call in the error
    -- system in the case of an exception.
    wf_core.context('<package_name>',
                    '<procedure_name>',
                    itemtype,
                    itemkey,
                    to_char(actid),
                    funcmode);
    raise;
end <procedure_name>;
```

Related Topics

Item Attributes, page 3-77

4

Defining a Workflow Process Diagram

This chapter tells you how to use Oracle Workflow Builder to define a workflow process diagram and how to load roles from the database so you can assign notification activities to specific roles.

This chapter covers the following topics:

- Process Window
- Diagramming a Process
- Modifying Fonts in Oracle Workflow Builder
- Creating a Shortcut Icon for a Workflow Process
- Referencing Roles
- Initiating a Process

Process Window

The process window in Oracle Workflow Builder graphically represents the activities (icons) and transitions (arrows) for a particular process. Each activity is a node, a logical step that contributes toward the completion of a process.

You can drag and drop activities from the navigator tree into the process window or create activities directly in the process window. The properties for an activity node may be viewed or edited by double clicking on the node in the process window with the select mouse button. You define transitions between activities by drawing arrows from one node to the next using the secondary mouse button.

Notification, function, event, and process activities make up the nodes of a process. If a process contains a process activity in its diagram, that process activity is known as a subprocess. There is no restriction on the depth of this hierarchy. To display the subprocess diagram in a process window, double-click on the subprocess activity node in the parent process window.

Transitions

Transitions appear as arrows in your diagram and represent the completion of one activity and the activation of another. For an activity that completes with a result type of <None>, any transition that you draw from it simply appears as an arrow to the next activity, indicating that as long as the originating activity completes, the process transitions to the next activity.

For an activity that has a defined result type, you must associate the transition arrow that you create with one of the activity's possible results. The result that the activity returns when it completes then determines what the next eligible activity is, as defined by the results-based transitions that originate from the completed activity. For example, "Notify Approver" with a result of 'REJECTED' transitions to "Reject Requisition." See: Requisition Process Activities, page 10-11.

You can also create a <Default>, <Any>, or <Timeout> transition for an activity that has a defined result type. The Workflow Engine follows a <Default> transition if no other transition matching the completion result exists. The Workflow Engine follows an <Any> transition regardless of what completion result the activity returns. This allows you to include a generic activity in the process that the Workflow Engine executes in parallel with the result-specific activity. The Workflow Engine follows a <Timeout> transition if the notification activity times out before completion. See: Setting Up Background Engines, *Oracle Workflow Administrator's Guide*.

Activities can have multiple transitions for a single result to create parallel branches.

Timeout Transitions

Draw a <Timeout> transition from a notification activity to some other activity to force the process to perform the other activity if the notification activity does not complete by a specified period of time. See: To Define Nodes in a Process, page 4-5.

When an activity times out, Oracle Workflow marks the activity as timed out and then cancels any notification associated with the timed out activity. The Notification System sends a cancellation message to the performer only if the canceled notification was expecting a response and the performer's notification preference is to receive e-mail.

> **Note:** If you are using the version of Oracle Workflow embedded in Oracle Applications, you can optionally use the WF: Mailer Cancellation Email profile option to prevent notification mailers from sending any notification cancellation messages. See: Setting Up Notification Mailers, *Oracle Workflow Administrator's Guide.*

Processing then continues along the <Timeout> transition as indicated by your process definition. If a timed out activity does not have a <Timeout> transition originating from it, Oracle Workflow executes the error process associated with the timed out activity or its parent process(es). See: To Define Optional Activity Details, page 3-67.

> **Note:** You must have a background engine set up to process timed out activities. See: Setting Up Background Engines, *Oracle Workflow Administrator's Guide*.

Creating Multiple Transitions to a Single Activity

You can create multiple transitions to a single activity in a process diagram. Sometimes these multiple transitions indicate that there are multiple ways that the process can transition to this one node and you may want the node to execute just once.

In other cases, the multiple transitions may indicate that the activity may be transitioned to multiple times because it is the starting point of a loop. In these cases, you want the activity to be reexecuted each time it is revisited.

The On Revisit flag for an activity determines whether the activity reexecutes when it is revisited more than once. It is an important flag to set for the pivot activity of a

loop. On Revisit is set initially in an activity's Details property page. However, for each usage of an activity in a process, you may change On Revisit for that node in the activity's Node property page. You can also use the standard Loop Counter activity as the initial activity in a loop to control how many times a process can transition through a loop. See: Looping, *Oracle Workflow API Reference* and Loop Counter Activity, page 5-5.

> **Tip:** If you have multiple incoming transitions from parallel branches, you should always use an And, Or, or custom join activity to merge those branches. This is especially true if after merging the parallel branches, you want to create a loop in your process. By using a joining activity to merge parallel branches and designating the following activity as the start of the loop, you create a less complicated process for the engine to execute. See: Standard Activities, page 5-1.

Designating Start and End Activities

Each process has to have a Start activity that identifies the beginning point of the process. You may designate any node from which it is logical to begin the process as a Start activity. When initiating a process, the Workflow engine begins at the Start activity with no IN transitions (no arrows pointing to the activity). If more than one Start activity qualifies, the engine runs each possible Start activity and transitions through the process until an End result is reached. The engine may execute acceptable Start activities in any order. Processes may contain multiple branches that each have an End activity. When the Workflow Engine reaches an End activity, the entire process ends even if there are parallel branches still in progress.

An End activity should return a result that represents the completion result of the process. The result is one of the possible values from that process activity's result type.

Start activities are marked with a small green arrow, and End activities by a red arrow, that appear in the lower right corner of the activity node's icon in the process window.

Diagramming a Process

This section discusses how to draw and define a workflow process in the process window:

- To Add Nodes to a Workflow Process, page 4-4
- To Define Nodes, page 4-5
- To Define Event Details for an Event Node, page 4-9
- To Define Activity Attribute Values, page 4-13
- To Create and Edit a Transition, page 4-14
- To Display a Process Overview, page 4-15
- To Print a Process, page 4-16
- To Copy a Process Diagram to the Clipboard, page 4-17
- To Validate a Process Definition, page 4-17

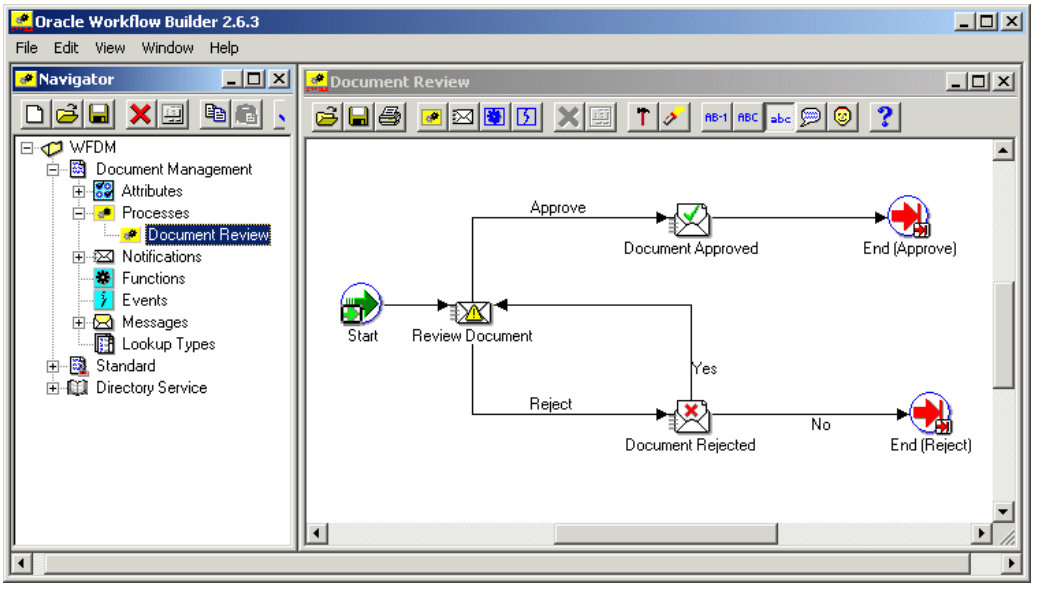
To Add Nodes to a Workflow Process:

1. To begin drawing a process diagram, you must first display the process window for your process activity. To display a process window, you can do one of several things:

   - Double-click on a predefined process activity on the navigator tree.
   - Select a predefined process activity and press Ctrl + E.
   - Select a predefined process activity and choose Process Details from the Edit menu.
   - Use the Quick Start Wizard to create a new process activity. See: To Use the Quick Start Wizard, page 2-14.

   A process window opens with the name of your process in the window title.

   Sample Oracle Workflow Builder Process Window



2. Create a new node in a process by using one of the following methods:

   - Drag and drop a notification, function, event, or process activity from the navigator tree into the process window. The activity you drag must belong to the same data store as the process you are dragging it to.

     Note: If you want to drag an activity into a process, where the activity is in a different data store than the process you are dragging it to, then you must first copy the item type that the activity belongs to into the same data store as the process.

   - Choose the New Function, New Process, New Event, or New Notification toolbar button to create a new activity. See: Process Window Toolbar, page B-6.
   - Choose New Function, New Process, New Event, or New Notification from the right mouse button menu while your cursor is in the process window to create a new activity node. An activity property page appears for you to select the activity for this node. See: To Define Nodes in a Process, page 4-5.

3. In the process window, you can display information about an activity by moving your mouse over the activity. The Label Name, Internal Name, Display Name, Comment and Performer, appear in a "tool-tip"-style display.
4. If you single click on an activity node in the process window, Oracle Workflow Builder expands the navigator tree and highlights the master activity of the node you select.
5. Create an arrow (transition) between two activity nodes by holding down your right mouse button and dragging your mouse from a source activity to destination activity.
6. If the source activity has no result code associated with it, then by default, no label appears on the transition. If you specifically choose to show the label for such a transition, the label <Default> appears. See: To Create and Edit a Transition, page 4-14.

   If the source activity has a result code associated with it, then a list of lookup values appears when you attempt to create a transition to the destination activity. Select a value to assign to the transition. You can also select the values <Default>, <Any>, or <Timeout> to define a transition to take if the activity returns a result that does not match the result of any other transition, if the activity returns any result, or if the activity times out, respectively.

   You can also drag and drop a lookup code from the navigator tree onto an existing transition in the process window to change the result of that transition. The lookup code you drag and drop must belong to the same data store and same lookup type as the lookup code you replace.
7. You can select an entire region of a process diagram, containing multiple activity nodes and transitions, and make a copy of the selection by holding down the Control or Shift key as you drag the selection to a new position in the process window.

   > **Important:** Oracle Workflow does not support reusing a subprocess activity multiple times within a process hierarchy. If you wish to use a subprocess more than once in a process, you must create a distinct copy of the subprocess for each instance needed.

8. Turn on grid snap from the View menu to snap your activity icons to the grid when you complete your diagram. Grid snap is initially turned on by default until you change the setting, at which point the latest setting becomes your default.

To Define Nodes in a Process:

1. Open the process window for your process activity.
2. To create a new function, notification, event, or process node, first select the New Function, New Notification, New Event, or New Process icon from the process window toolbar. Next, click on the position within the process window where you want to place this new node. The property page for the new node appears.

   > **Note:** You can also create a new node by dragging and dropping a predefined activity from the navigator tree into the process window. This automatically populates the node's property page with predefined information. Double-click on the node and skip to Step 5 to further edit its property page.

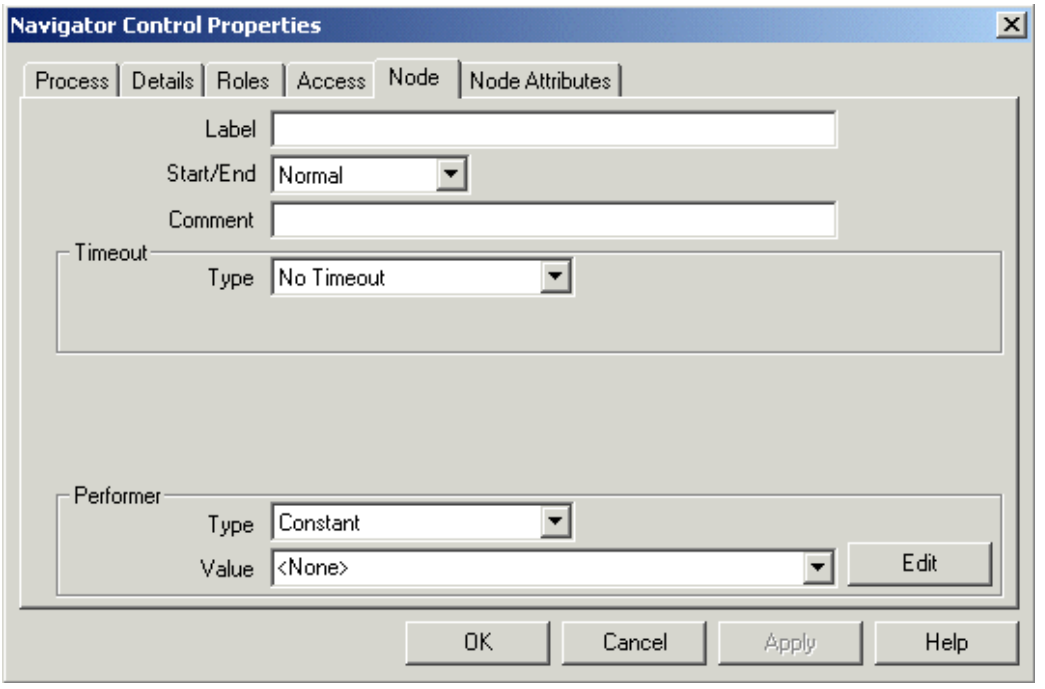
3. In the Item Type field, select the item type that you want this activity node to be associated with.
4. Choose one of the following methods to define the remaining information for the node.
   - Select either the internal name or display name of a predefined activity. Oracle Workflow Builder then populates all the fields with predefined information from the master activity as shown in the Navigator window.
   - Alternatively, choose the New button to define a new activity. To complete the following tabs of the property page, refer to the sections listed:
     - Process - To Create a Process Activity, page 3-66
     - Function - To Create a Function Activity, page 3-60
     - Notification - To Create a Notification Activity, page 3-58
     - Event - To Create an Event Activity, page 3-62
     - Details - To Define Optional Activity Details, page 3-67
     - Roles - The information in this tab is currently not supported.
     - Access - To Set the Access Level for an Object, page 3-18

       Caution: Any changes that you make to the any of the above tabs automatically propagate to the master activity and affect all other instances of that activity. Only changes that you make to the Node and Node Attributes tabs, and to the Event Details tab for an event activity, are local and specific to the current node activity.

5. Select the Node tab to specify information that is specific to this node. Specify a Label for the node. Since an activity can be used more than once in any given process, the Label field lets you give a unique name to the instance of this particular activity in the process. By default, the label name is the activity name, but if the activity is used more than once in the process, `-N` is appended to the activity name, where `N` represents the '*N*th' instance of the activity in use.

   Important: When you call most Oracle Workflow APIs, you must pass the activity's label name and not its activity name. See: Workflow Engine APIs, *Oracle Workflow API Reference*.

Node Tab



6. Indicate if the current node is a start or end activity in your process, by choosing 'START' or 'End', respectively. 'NORMAL' is the default if it is neither. You may have multiple START and END nodes in your process.

   A Start activity is marked `(Start)` and has a small green arrow in its activity icon, and an End activity is marked `(End)` and has a red arrow in its activity icon.

   > **Important:** The Start/End field is always set to Normal by default for all activity nodes. Even if you use the Standard Start or Standard End activity, you must manually edit the Start/End field to be either Start or End, respectively.

7. For an END node, you must also select a value for the final process result if the overall process activity has a result type associated with it. The list of values for the final process result is derived from the lookup type specified as the process activity's result type.

8. You can provide a comment to yourself about this node.

9. For a notification that requires a response, a process activity that is a subprocess within another process, an event activity with an event action of Receive, or a blocking function activity, specify whether the activity must be completed by some specified time. If the activity is not completed by a given time, you can redirect the parent process to transition to a different activity. See: Timeout Transitions, page 4-2.

   Choose 'No Timeout' if the activity does not have to be completed by a given time.

   Choose 'Relative Time' if you want the activity to be completed by some constant relative time. You can enter any combination of days, hours and minutes to specify when the activity times out. The value you enter is interpreted as a relative offset from the begin date of the activity, in the unit of minutes. A relative timeout value of zero means no timeout.

Choose 'Item Attribute' if you want the activity to be completed by some relative time that is computed dynamically at runtime. Note that you must first create an item attribute of type number to store the computed timeout value and reference that predefined item attribute here. See: Item Type Attributes, page 3-2 and To Define an Item Type or Activity Attribute, page 3-7.

> **Important:** The dynamic timeout value stored in this attribute is interpreted as a relative offset from the begin date of the activity, in the unit of minutes. A null timeout value or a value of zero means no timeout.

10. For a notification activity node, or for an event activity node with an event action of Send, you can override the priority assigned to the activity's message. Choose 'Default' to keep the default priority of the message.

    Choose 'Constant' to override the default priority with the new specified priority level.

    Choose 'Item Attribute' to override the default priority with a new priority level that is dynamically determined at runtime. Note that you must first create an item attribute of type number to store the computed priority value and reference that predefined item attribute here. See: Item Type Attributes, page 3-2 and To Define an Item Type or Activity Attribute, page 3-7.

    > **Note:** The computed priority value can be any number between 1-99. Oracle Workflow automatically converts the number to a priority level as follows: 1-33 = High, 34-66=Normal, and 67-99=Low.

11. For a notification activity node, specify the performer of the activity. The performer is the role to whom the notification is sent. You may either select a constant role name or an item type attribute that dynamically determines the role at runtime. Note that you must first create an item attribute of type role to store the computed role name and reference that predefined item attribute here. See: Item Type Attributes, page 3-2, To Define an Item Type or Activity Attribute, page 3-7, and Referencing Roles, page 4-19.

    > **Note:** If you set the Performer Type to Constant and you are connected to the database and have loaded roles from the database, you can select a constant role name from the Performer poplist. If you are working in a `.wft` file data store without any open connection to the database, you can directly type in a valid role display name in the Performer field. When you upload the file to a database, the role will be resolved to the appropriate role data stored in the database based on the role display name you entered.

    > **Note:** When you assign a notification to a multi-user role, the Workflow Engine keeps track of the individual from that role that actually responds to the notification. See: Respond API, *Oracle Workflow API Reference*.

    > **Note:** Although Oracle Workflow Builder allows you to specify a performer for any type of node activity, Oracle Workflow only considers the value of Performer for notification activity nodes.

12. Choose Apply to save your changes, OK to save your changes and close the property page or Cancel to cancel your changes and close the property page.

    When you save and close your property page, the activity node appears in the position you specified in the process window. If this is a new activity you created, a corresponding master activity is also created under the appropriate branch in the navigator tree.

13. If the node is an event activity, you can specify additional required event information by choosing the Event Details tab. See: To Define Event Details for an Event Node, page 4-9.

14. If the node is a function, notification, or event activity and the activity has activity attributes, you can assign values to those activity attributes by choosing the Node Attributes tab. See: To Define Activity Attribute Values, page 4-13.

15. If the node is a process activity, then a small subprocess overlay icon appears over the upper right corner of process activity icon. The subprocess overlay icon identifies the node as a subprocess within the process diagram.

To Define Event Details for an Event Node:

The event action defined for the event activity determines which event details you must define for an event node. For each event detail, it is either required or optional to use an item type attribute to store or retrieve the detail information. Note that you must first create item type attributes of the appropriate types before you can reference those predefined item attributes here. The item type attributes you use for event details must be associated with the same item type as the event activity itself. See: Item Type Attributes, page 3-2, To Define an Item Type or Activity Attribute, page 3-7, Event Activity, page 3-54, and To Create an Event Activity, page 3-62.

1. Display the property pages of an event activity node. Select the Event Details tab.

2. For an activity with the event action Receive, enter the following event details:

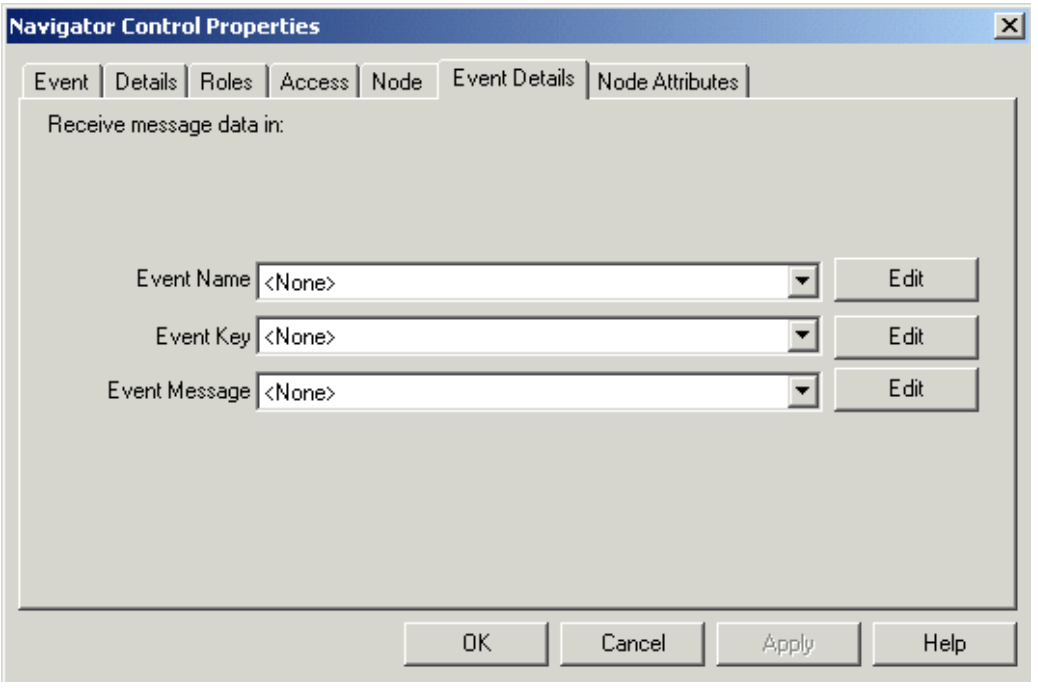
   - Event Name - Optionally select an item type attribute of type text where you want to store the event name that the node receives.

     > **Note:** The event activity can only receive events that match its event filter. Additionally, if the event activity has a `#BUSINESS_KEY` attribute, it can only receive an event whose event key matches its `#BUSINESS_KEY` value. See: To Create an Event Activity, page 3-62.

   - Event Key - Optionally select an item type attribute of type text where you want to store the event key that the node receives.

   - Event Message - Optionally select an item type attribute of type event where you want to store the event message that the node receives.

Event Details Tab for a Receive Event Activity



> **Note:** When the activity receives an event, the Workflow Engine stores the event name, event key, and event message in the item type attributes you specify, and also sets any parameters in the event message parameter list as item type attributes for the process, creating new item type attributes if a corresponding attribute does not already exist for any parameter.
>
> Additionally, if the event was originally raised by a Raise event activity in another workflow process, the item type and item key for that process are included in the parameter list within the event message. In this case, the Workflow Engine automatically sets the specified process as the parent for the process that receives the event, overriding any existing parent setting. See: SetItemParent, *Oracle Workflow API Reference*.

3. For an activity with the event action Raise, enter the following event details:
   - Event Name - Enter the name of the event that the node raises. You can either specify a constant event name or select an item type attribute of type text that dynamically determines the event name at runtime.

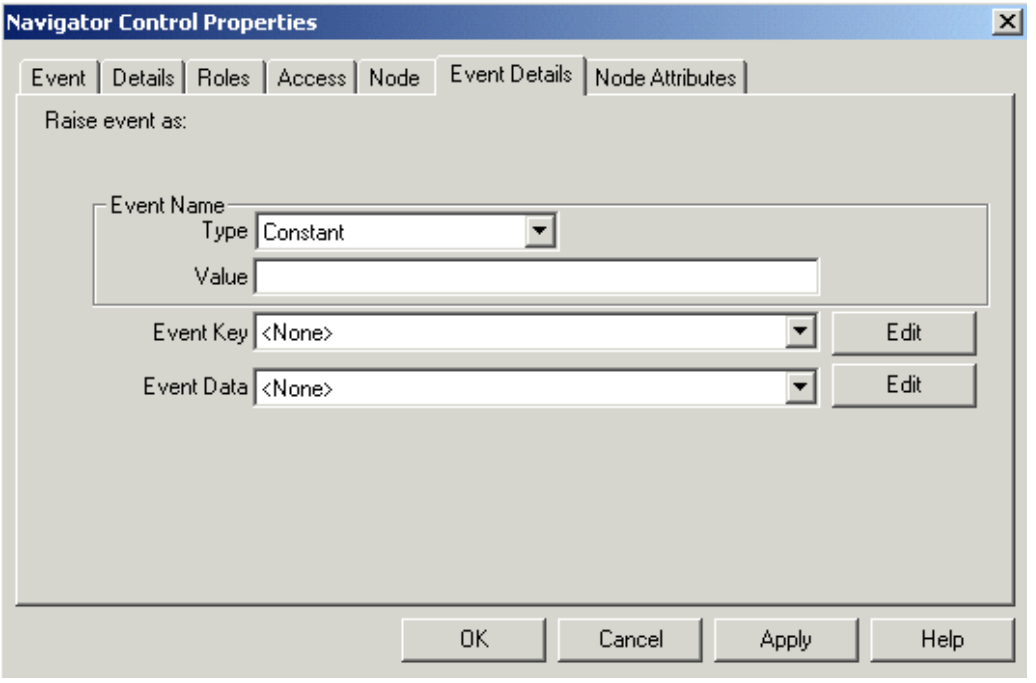
     > **Note:** You can only raise an individual event. You cannot raise event groups.

   - Event Key - Select the item type attribute of type text that contains the event key for the event that the node raises.
   - Event Data - Optionally select an item type attribute that contains the event data for the event that the node raises. You can store event data in item type attributes of type text, number, date, lookup, role, or attribute. The maximum

length of the data you can enter in a text attribute is 4000 bytes. If the event data exceeds 4000 bytes, you should assign a generate function in the event definition to generate the event data, rather than providing the event data through a text attribute. See: To Define an Event, page 8-5.

You must not specify an item type attribute of type event for the Event Data field, since the event data is only a part of the complete event message structure which is the format for the event attribute type. If you want to use the event data stored within an existing event message, leave the Event Data field unspecified and instead define the `#EVENTMESSAGE2` activity attribute for the raise event activity.

Event Details Tab for a Raise Event Activity



Note: The Event Name and Event Key are required for a Raise event activity.

Note: In addition to these event details, you can use the activity attributes for a Raise event activity node to specify parameters that you want to include in the parameter list for the event message. If the event message is later received by another process, the Workflow Engine sets the event parameters as item type attributes for that process. See: To Define Activity Attribute Values, page 4-13.

Also, a Raise event activity automatically sets the item type and item key for the current workflow process in the parameter list for the event message. If the event message is later received by another process, the Workflow Engine uses that item type and item key to automatically set the process that raised the event as the parent for the process that receives the event. See: SetItemParent, *Oracle Workflow API Reference*.

If you want to raise the new event using the event data and parameter list from an existing event message, you can also define a special activity attribute named `#EVENTMESSAGE2` for the raise event activity. Set the existing event message as the value of the `#EVENTMESSAGE2` attribute, which must be an attribute of type event. If this attribute is defined, the activity retrieves the event data and parameter list from the specified event and sets them into the new event message before it is raised.

If you also specified event data in the node's event details, however, the activity sets that event data into the event, overriding any event data from the `#EVENTMESSAGE2` attribute. If you specified any additional parameters in activity attributes for the raise event activity, the activity also sets those parameters into the parameter list for the event message, overriding the values of any parameters with the same names from the `#EVENTMESSAGE2` attribute.

4. For an activity with the event action Send, enter the following event details:
   - Event Message - Select the item type attribute of type event that contains the event message that the node sends.
   - Event Name - Optionally enter the name of the event that the node sends. You can either specify a constant event name or select an item type attribute of type text that dynamically determines the event name at runtime. The event name that you enter here overrides the previous event name value in the event message.
   - Event Key - Optionally select an item type attribute of type text that contains the event key of the event that the node sends. The event key that you enter here overrides the previous event key value in the event message.
   - Out Agent - Optionally enter the outbound agent from which the node sends the event. Specify both the agent name and the system name for the agent using the following format:

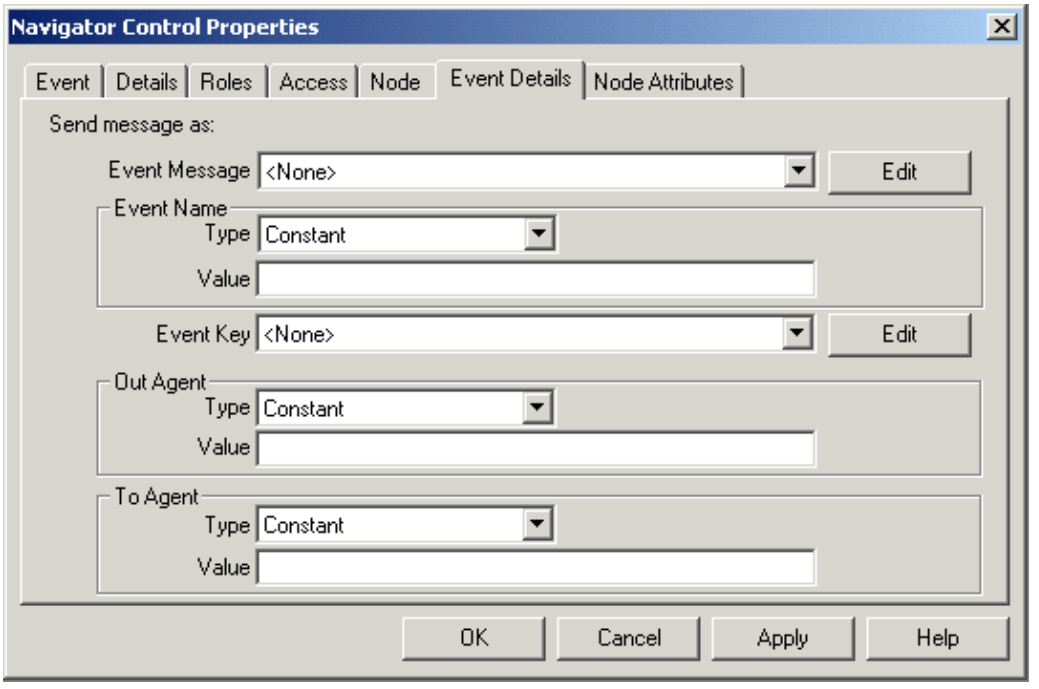
     ```
     <agent_name>@<system_name>
     ```

     You can either specify a constant Out Agent name or select an item type attribute of type text that dynamically determines the Out Agent name at runtime. The Out Agent that you enter here overrides the previous outbound agent value in the event message.
   - To Agent - Optionally enter the inbound agent to which the node sends the event. Specify both the agent name and the system name for the agent using the following format:

     ```
     <agent_name>@<system_name>
     ```

     You can either specify a constant To Agent name or select an item type attribute of type text that dynamically determines the To Agent name at runtime. The To Agent that you enter here overrides the previous inbound agent value in the event message.

Event Details Tab for a Send Event Activity



Note: The Event Message is required for a Send event activity. Additionally, you must either include a To Agent or a From Agent within the event message, or specify a To Agent or an Out Agent in the event details for this node. If you neither specify an inbound agent nor an outbound agent, the event cannot be sent.

Note: If no correlation ID is initially specified in the event message, Oracle Workflow automatically sets the correlation ID to the item key of the process.

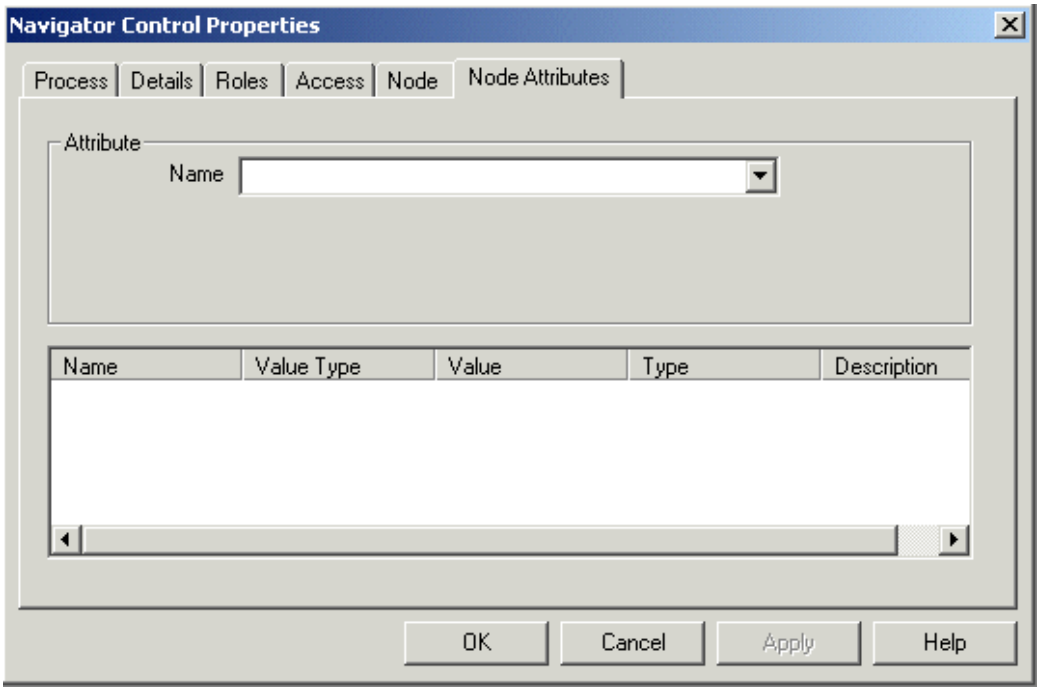
5. Choose Apply to save your changes, OK to save your changes and close the property page, or Cancel to cancel your changes and close the property page.

To Define Activity Attribute Values:

Activity attribute values for a function or notification activity are used by the PL/SQL stored procedure that the activity calls. Activity attribute values for a raise event activity are set as parameters in the parameter list for the event message, except for the special `#EVENTMESSAGE2` activity attribute. If the `#EVENTMESSAGE2` attribute is defined, the raise event activity retrieves the event data and parameter list from the specified existing event and sets them into the new event message before it is raised. For a receive event activity, the special `#BUSINESS_KEY` attribute is used to determine whether the activity is eligible to receive an event sent from the Event Manager based on whether the business key value matches the event key. See: To Define an Item Type or Activity Attribute, page 3-7.

1. Display the property pages of an activity node. Select the Node Attributes tab.

Node Attributes Tab



2. Select an attribute.
3. In the Value region, enter the value for this attribute. The value can be a constant or a value stored in an item type attribute.

   The value you enter must match the data type of the activity attribute, and of the actual activity parameter itself as it is defined in the PL/SQL function associated with the activity, if there is one. The attribute type is displayed along with the name, description, value type, and value of each attribute in the attributes summary region.
4. Choose Apply to save your changes, OK to save your changes and close the property page or Cancel to cancel your changes and close the property page.

To Create and Edit a Transition:

1. To create a transition between two activities, hold down your right mouse button and drag your mouse from a source activity to a destination activity.

   Note: Overlapping transitions appear in a different color than single, non-overlapping transitions.

2. To edit a transition, select the transition.
3. To reposition a transition label, simply select the label with your mouse and drag it to its new position. The label snaps onto the transition.
4. You can bring up the following menu of editing options at any time by selecting a transition with your mouse and clicking on the right mouse button:
   - Delete Transition - deletes the selected transition.

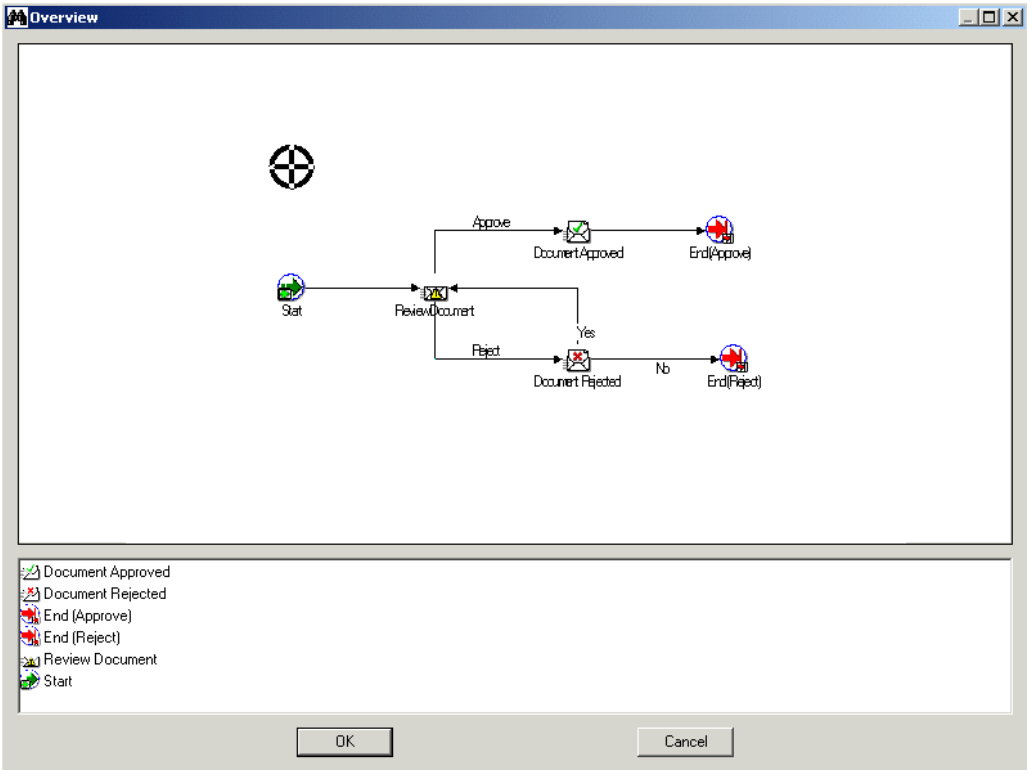
- Locked - toggles between locking and unlocking the transition from further edits. If a transition is locked, you cannot add or delete vertex points along the transition, but you can delete the transition.
- Hidden Label - toggles between displaying and hiding the transition label.
- Straighten - straightens the transition by removing the extra vertex points that can cause bends in the transition.
- Results... - if the transition has a result assigned to it, use this option to change the result label on the transition. An additional menu appears that lists the possible result labels you can choose.

5. To bend a transition, create a vertex point by selecting the transition and dragging the transition as you hold down your left mouse button. You can reposition any vertex point to create a bend in the transition.
6. You can create a transition that loops back to its source activity node in one of two ways:
   - Hold down your right mouse button and drag your mouse from a source activity back to itself to create a self loop.
   - From a source activity node, create a transition to another arbitrary activity node. Add a vertex point to create a bend in the transition. Then select and drag the arrowhead of the transition back to the source activity node. Create additional vertex points as necessary to improve the visual display of the looping transition.
7. To remove a single vertex point from a transition, select the vertex and drag it over another vertex to combine the two points.

To Display a Process Overview:

1. Place your cursor in the process window and choose Overview from the right mouse button menu.
2. An Overview dialog window of your process appears.

Overview Window



The upper pane of the window shows a size-reduced sketch of your entire process, while the bottom pane is a list of the activities in your process.

3. You can resize the Overview dialog window to get a better view of the process sketch.
4. A cross hairs cursor that you can drag appears in the process sketch pane. Use the cross hairs cursor to pinpoint an area in your process that you want the process window to display.
5. Single click on an activity in the lower pane to move the cross hairs cursor to that activity within the sketch. Choose OK to close the dialog window and to jump to that activity in the process window.

   You can also drag and double-click on the cross hairs cursor in the upper pane to close the dialog window and to jump to the resulting region in the process window.

To Print a Process:

1. Display the process window containing the process you wish to print.
2. With the process window selected as the active window, choose Print Diagram from the File menu or from the right mouse button menu.

   The Print Diagram option captures your process diagram as a picture (metafile), enlarges it to the correct size to print and sends it to a printer. If your diagram is large, it may span more that one page when printed. However, depending on the printer driver you use, you may get a Print dialog box that lets you scale your image down to one page for printing.

Note: If your process diagram uses a font that the printer cannot find, your printer driver may either substitute a similar font or not print any text.

To Copy a Process Diagram to the Clipboard:

1. Display and make the process window containing the process you wish to copy active.
2. Choose Copy Design from the Edit menu or from the right mouse button menu.

   This copies the process to the clipboard in the form of a metafile and a bitmap diagram.
3. To paste the metafile-version or bitmap-version of the process diagram into another application window, you should consult the other application's documentation on how to paste metafiles or bitmaps.

   To edit a bitmap image, you must paste the image into an application that can edit bitmaps.

To Validate a Process Definition:

1. Choose Verify from the File menu to validate all process definitions for the currently selected data store.
2. The following list is an example of some of the validation that the Verify command performs:
   - Checks that a process has at least one Start and one End activity.
   - Verifies that a process does not contain itself as a process activity.
   - Restricts the same subprocess from being used twice in a process.
   - Validates that all possible activity results are modelled as outgoing transitions. If an activity completes with a result that is not associated with an outgoing transition, and a <Default> transition doesn't exist for that activity, the activity enters an 'ERROR' state.
   - Validates that activity nodes marked as END nodes do not have any outgoing transitions.
   - Validates that a notification activity's result type matches the lookup type defined for the message's 'RESULT' message attribute.
   - Verifies that message attributes referenced in a message body for token substitution exist in the message definition.
   - For processes that reference objects from another item type, verifies that the requisite item attributes associated with the referenced item type exist.

     Important: You should always validate any new process definition you create as it helps you to identify any potential problems with the definition that might prevent it from executing successfully.

Related Topics

To Find an Object in the Navigator Tree, page 2-4

Using the Edit Button in a Property Page, page 2-6

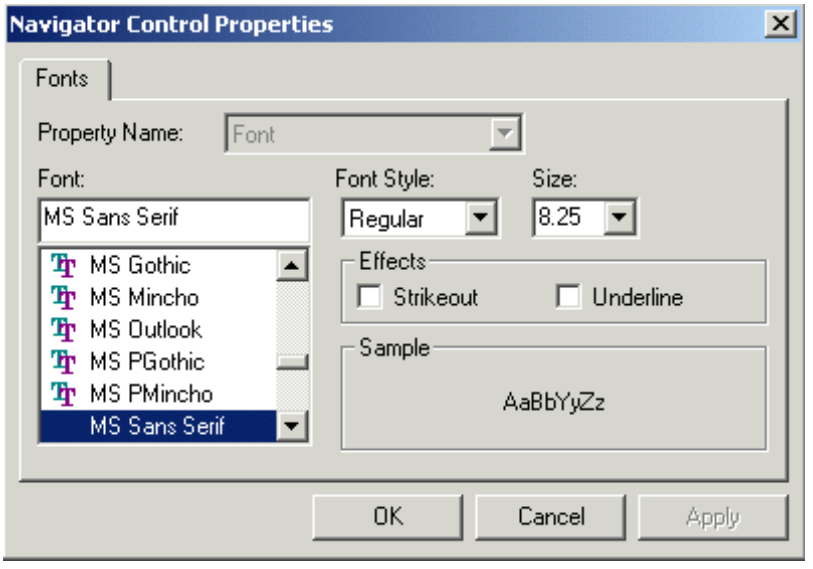
Modifying Fonts in Oracle Workflow Builder

You can modify the font that is used by the windows in Oracle Workflow Builder. Any change you make applies to all windows within the program.

To Modify Fonts:

1. Choose Font from the View menu to display the Fonts property page.

Fonts Property Page



2. Select the font to use as the label for your icons. This font is used for all icons in Workflow Builder. The Sample region shows the appearance of the font you select.
3. Choose the font style: Regular, Bold, Italic or Bold Italic. Some fonts have a limited selection of font styles.
4. Indicate the font size to use. Some fonts have a limited selection of font sizes.
5. Select the Underline or Strikeout check boxes to apply that effect.
6. Choose OK when you are done. These font settings take effect immediately and are also used the next time you start Oracle Workflow Builder.

Creating a Shortcut Icon for a Workflow Process

You can create a shortcut to Oracle Workflow Builder on your Windows desktop. The shortcut can start Oracle Workflow Builder by automatically connecting to a designated data store and opening specific process windows from that data store.

To Create an Oracle Workflow Builder Shortcut:

1. Start Oracle Workflow Builder.
2. Choose Open from the File menu to open a data store.
3. Optionally expand the Process branch and double-click on one or more process activities to open the process windows for those processes.

4. Choose Create Shortcut from the File menu.
5. Enter a name for the shortcut, as you want it to appear on your desktop.
6. When you double-click on the new shortcut icon on your desktop, it automatically starts Oracle Workflow Builder opening the data store that was selected and any process windows that were open when you created the shortcut.

   If the data store for the shortcut is a database, the shortcut will prompt you for the password to the database.

Referencing Roles

Oracle Workflow roles are stored in the database, in the Oracle Workflow directory service. Currently, new workflow roles cannot be created in Oracle Workflow Builder, but Oracle Workflow Builder can display and reference the roles stored in a database.

One example of how roles are referenced in a workflow process is when you include a notification activity in a process as a node. You must assign that node to a performer. The performer can be a designated role or an item type attribute that dynamically returns a role. To assign a performer to a role, you must initially load the roles from your Oracle Workflow database into your Oracle Workflow Builder session. See: Setting Up an Oracle Workflow Directory Service, *Oracle Workflow Administrator's Guide* and To Define Nodes in a Process, page 4-5.

> **Note:** Referencing roles in a workflow process is currently supported in Oracle Workflow Builder, although the Roles tab page seen in the property pages of certain workflow objects will not be supported until a future release. The purpose of the Roles tab page is to give a role access to a certain object.

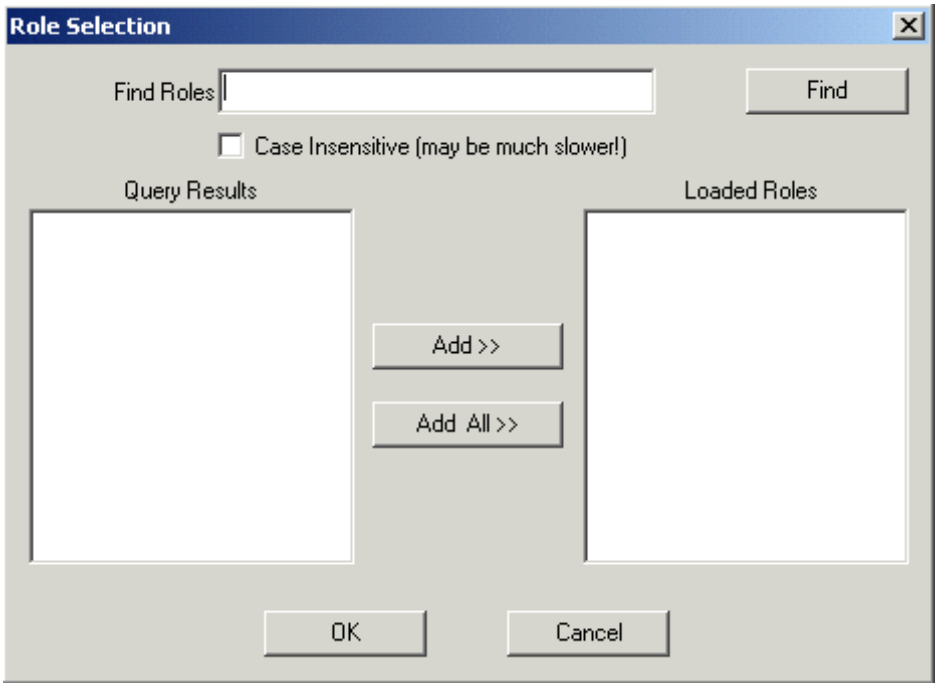
Ad Hoc Users and Roles:

Oracle Workflow allows you to create new ad hoc users and roles within a workflow process, to add to your directory service. To do so, you define a function activity that makes a server-side call to the appropriate WF_DIRECTORY API and include that function activity in your process diagram. See: Standard API for PL/SQL Procedures Called by Function Activities, page 6-2 and Workflow Directory Service APIs, *Oracle Workflow API Reference*.

To Load Roles:

1. If you are not connected to an Oracle Workflow database, choose Open from the File menu to connect to the database and open your item type.
2. Choose from the File menu, Load Roles from Database. A Role Selection window appears. You can enter search criteria using SQL query syntax in the Find Roles field to find a subset of roles, or just choose Find without specifying any search criteria to identify all roles. The Role Selection window finds the roles you specify and displays them in the Query Results list box.

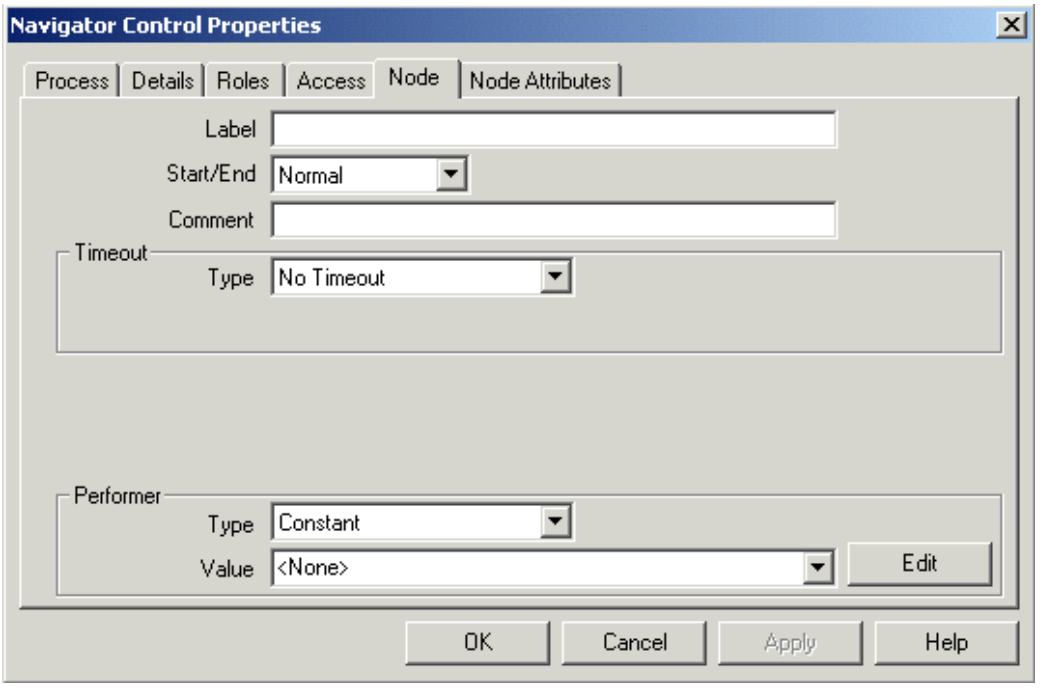
Role Selection Window



3. Select the roles you want to load from the Query Results list and choose Add to add them to the Loaded Roles list. Alternatively, just choose Add All to add all the roles in the Query Results list to the Loaded Roles list. Choose OK to load the selected roles into Oracle Workflow Builder and make them available to the workflow objects in your open item type.

   The workflow objects that need to reference role information contain specific fields in their property pages. These fields are poplist fields that display the list of roles you loaded from the database, such as the Performer fields in the Node property page.

Node Property Page



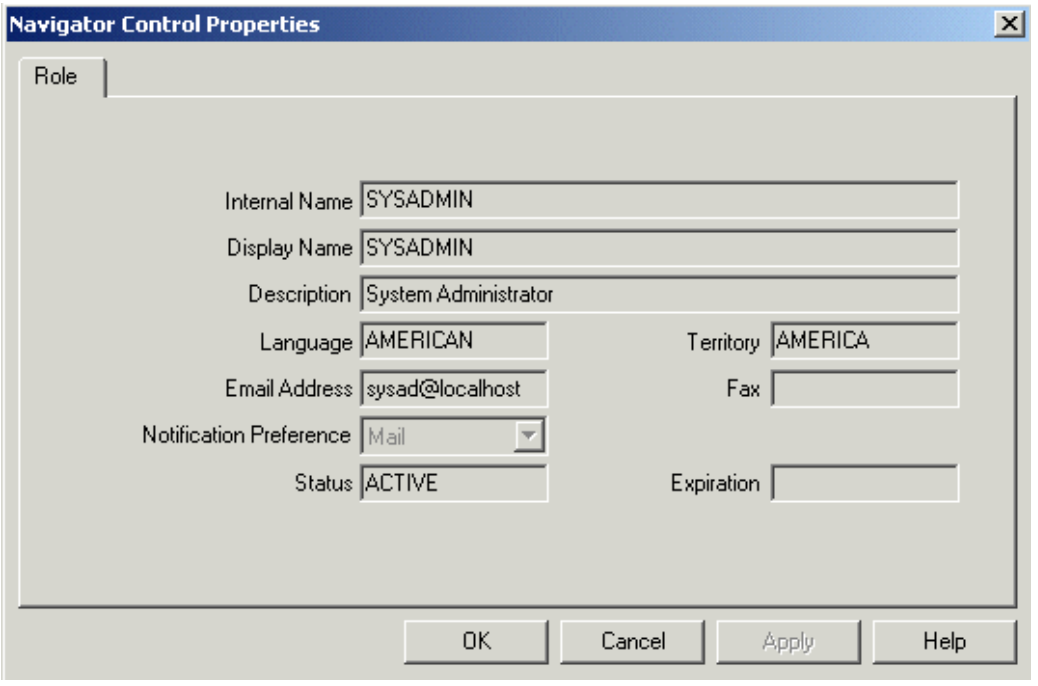
4. When you select a role from one of these poplist fields, you can also choose the Edit button to the right of the field to display the property sheet of the selected role.
5. The Role property page that appears lists read-only information about that role.

   Note: When you reopen a saved process definition in Oracle Workflow Builder, any role information that the process references automatically gets loaded even if you open the process definition from a file and are not connected to the database.

To Display the Directory Service in Oracle Workflow Builder:

1. Once you load your roles from the database in Oracle Workflow Builder, you can display your directory service information in the navigator tree. See: To Load Roles, page 4-19.
2. Expand the Directory Service branch in the navigator tree. All the roles that you loaded from the database appear.
3. Double-click on a role to display read-only information about that role, including the role's internal name, display name, description, language, territory, e-mail address, fax number, notification preference, status, and expiration date. Note that the Directory Service branch does not currently allow you to view the participant users of a role.

Role Property Page



Initiating a Process

A workflow process begins when an application calls the Workflow Engine *CreateProcess()* and *StartProcess()* APIs or when a Business Event System subscription sends an event to launch the process. A subprocess is started when the Workflow Engine transitions to a process activity that represents the subprocess.

To launch a workflow process using the Business Event System:

1. Define a business event.
2. Define a subscription to this business event. In the subscription properties, specify the workflow item type and process that you want to launch.

   By default, Oracle Workflow uses the event key as the item key for the workflow process that is launched. If you want to generate the item key based on a custom rule, create a function that populates the correlation ID in the event message with the item key you want, and assign that function as the subscription's rule function.

   > **Note:** The item key for a process instance can only contain single-byte characters. It cannot contain a multibyte value.

3. Add the Raise() API to your custom application code at the point where you want to launch the workflow process.

Related Topics

Workflow Engine APIs, *Oracle Workflow API Reference*

Raise, *Oracle Workflow API Reference*

Managing Business Events, page 8-2

5

Predefined Workflow Activities

This chapter tells you how to use Oracle Workflow's predefined activities.

This chapter covers the following topics:

- Standard Activities
- Concurrent Manager Standard Activities

Standard Activities

Oracle Workflow provides some generic activities you can use to control your process. The activities are associated with the Standard item type but can be used within any process you define. The Standard item type is automatically installed on your Oracle Workflow server. You can also access the Standard item type from the file `wfstd.wft` located on your PC in the `<ORACLE_HOME>\Wf\data\<language>` directory.

> **Note:** Predefined activities are also available for the predefined workflows shipped with Oracle Applications. For more information on Oracle Applications-specific workflow activities, consult the documentation or help for that specific Oracle Applications product.

> **Note:** If you want to drag an activity into a process, where the activity is in a different data store than the process you are dragging it to, then you must first copy the item type that the activity belongs to into the same data store as the process. Suppose you are modifying a process that is stored in `wfexample.wft` and you want to add some standard activities into the process that are stored in `wfstd.wft`. First you need to open both files as data stores in Oracle Workflow Builder, then you need to copy the Standard item type in `wfstd` and paste it into the `wfexample` data store. Now you can drag any standard activity in the `wfexample` data store into your process.

Related Topics

And/Or Activities, page 5-2

Comparison Activities, page 5-2

Compare Execution Time Activity, page 5-3

Wait Activity, page 5-3

Block Activity, page 5-4

Defer Thread Activity, page 5-4

Launch Process Activity, page 5-5

Noop Activity, page 5-5

Loop Counter Activity, page 5-5

Start Activity, page 5-7

End Activity, page 5-7

Role Resolution Activity, page 5-7

Notify Activity, page 5-8

Vote Yes/No Activity, page 5-8

Master/Detail Coordination Activities, page 5-9

Assign Activity, page 5-12

Get Monitor URL Activity, page 5-12

Get Event Property Activity, page 5-12

Set Event Property Activity, page 5-13

Compare Event Property Activity, page 5-13

XML Get Tag Value Activity, page 5-14

XML Compare Tag Value Activities, page 5-15

XML Transform Activity, page 5-16

And/Or Activities

In cases where multiple parallel branches transition to a single node, you can decide whether that node should transition forward when *any* of those parallel branches complete or when *all* of the parallel branches complete. Use the And activity as the node for several converging branches to ensure that all branches complete before continuing. Use the Or activity as the node for several converging branches to allow the process to continue whenever any one of the branches completes.

- **And** - Completes when the activities from all converging branches complete. Calls a PL/SQL procedure named *WF_STANDARD.ANDJOIN*.
- **Or** - Completes when the activities from at least one converging branch complete. Calls a PL/SQL procedure named *WF_STANDARD.ORJOIN*.

Comparison Activities

The comparison activities provide a standard way to compare two numbers, dates, or text strings.

- **Compare Date** - Use to compare the value of an item type attribute of type Date with a constant date.
- **Compare Number** - Use to compare the value of an item type attribute of type Number with a constant number.
- **Compare Text** - Use to compare the value of two item type attributes of type Text.

All the Comparison activities call a PL/SQL procedure named *WF_STANDARD.COMPARE*.

Activity Attributes

Each comparison activity has two activity attributes:

- Test Value - a constant number, date, or text string which to compare to a reference value.
- Reference Value - an item type attribute of type Number, Date, or Text.

The comparison activities use the Comparison lookup type for a result code. Possible values are "Greater Than," "Less Than," "Equal," or "Null," if the item type attribute is null. You can guide your workflow process based on how the value of an item type attribute compares to a given value that you set. See: To Define Activity Attribute Values, page 4-13.

Compare Execution Time Activity

The Compare Execution Time activity provides a standard way to compare the elapsed execution time of a process with a constant test time.

The Compare Execution Time activity calls a PL/SQL procedure named *WF_STANDARD.COMPAREEXECUTIONTIME*.

Activity Attributes

The Compare Execution Time activity has two activity attributes:

- Test Execution Time - the time, in seconds with which to compare the elapsed execution time.
- Parent Type - takes as its value, the lookup codes, "Root" or "Parent". A value of "Root" compares the test time with the elapsed execution time of the current root process. A value of "Parent" compares the test time with the elapsed execution time of just the immediate parent process, which can be a subprocess.

The activity uses the Comparison lookup type for a result code. Possible values are "Greater Than," "Less Than," "Equal," or "Null," if the test time is null. See: To Define Activity Attribute Values, page 4-13.

Wait Activity

The Wait activity pauses the process for the time you specify. You can either wait until:

- a specific date
- a given day of the month
- a given day of the week
- a period of time after this activity is encountered

This activity calls the PL/SQL procedure named *WF_STANDARD.WAIT*.

> **Note:** You must run a background engine for deferred activities to determine when the wait time has passed. The background engine then completes the Wait activity so that the process can continue. See: Setting Up Background Engines, *Oracle Workflow Administrator's Guide*.

Activity Attributes

The Wait activity has six activity attributes:

- Wait Mode - use this attribute to specify how to calculate the wait. You can choose one of the following wait modes:
  - Absolute Date - to pause the activity until the date specified in the Absolute Date activity attribute is reached.
  - Relative Time - to pause the activity until the number of days specified in the Relative Time activity attribute passes.
  - Day of Month - to pause the activity until a specified day of the month, as indicated in the Day of Month activity attribute.
  - Day of Week - to pause the activity until a specified day of the week, as indicated in the Day of Week activity attribute.
- Absolute Date - If Wait Mode is set to Absolute Date, enter an absolute date.
- Relative Time - If Wait Mode is set to Relative Time, enter a relative time expressed in `<days>.<fraction of days>`. For example, enter `0.5` for a wait time of half a day (12 hours).
- Day of Month - If Wait Mode is set to Day of Month, choose a day of the month from the list. If the day you choose has already passed in the current month, then the activity waits until that day in the following month.
- Day of Week - If Wait Mode is set to Day of Week, choose a day of the week from the list. If the day you choose has already passed in the current week, then the activity waits until that day in the following week.
- Time of Day - The Wait activity always pauses until midnight of the time specified, unless you use this Time of Day activity attribute to specify a time other than midnight that the Wait activity should pause until.

See: To Define Activity Attribute Values, page 4-13.

Block Activity

The Block activity lets you pause a process until some external program or manual step completes and makes a call to the *CompleteActivity* Workflow Engine API. Use the Block activity to delay a process until some condition is met, such as the completion of a concurrent program. Make sure your program issues a *CompleteActivity* call when it completes to resume the process at the Block activity. See: CompleteActivity, *Oracle Workflow API Reference*.

This activity calls the PL/SQL procedure named *WF_STANDARD.BLOCK*.

Defer Thread Activity

The Defer Thread activity defers the subsequent process thread to the background queue without requiring you to change the cost of each activity in that thread to a value above the Workflow Engine threshold. This activity always interrupts the process thread by causing a disconnect to occur in the current database session, even if the thread is already deferred.

This activity calls the PL/SQL procedure named *WF_STANDARD.DEFER*.

Launch Process Activity

The Launch Process activity lets you launch another workflow process from the current process. This activity calls the PL/SQL procedure named *WF_STANDARD.LAUNCHPROCESS*.

Activity Attributes

The Launch Process activity has six activity attributes:

- Item Type - the item type of the process to launch. Specify the item type's internal name. This activity attribute requires a value.
- Item Key - an item key for the process to launch. If you do not specify a value, the item key defaults to `<current_item_type>:<current_item_key>-<n>`, where `<current_item_type>` and `<current_item_key>` identify the current process instance, and `<n>` is the number of processes launched by the current process instance, starting at 1.

  Note: The item key for a process instance can only contain single-byte characters. It cannot contain a multibyte value.

- Process name - the internal name of the process to launch. If a process name is not specified, the activity will check the item type selector function of the process to launch for a process name.
- User Key - a user defined key for the process to launch.
- Owner - a role designated as the owner of the process to launch.
- Defer immediate - choose between `YES` or `NO` to determine whether the process to launch should be immediately deferred to the background engine. The default is `NO`, so once the process is launched, it continues to execute until completion or until one of its activities is deferred.

See: To Define Activity Attribute Values, page 4-13.

Noop Activity

The Noop activity acts as a place holder activity that performs no action. You can use this activity anywhere you want to place a node without performing an action. You can change the display name of this activity to something meaningful when you include it in a process, so that it reminds you of what you want this activity to do in the future. This activity calls the PL/SQL procedure named *WF_STANDARD.NOOP*.

Loop Counter Activity

Use the Loop Counter activity to limit the number of times the Workflow Engine transitions through a particular path in a process. The Loop Counter activity can have a result of Loop or Exit.

This Loop Counter activity calls the PL/SQL procedure named *WF_STANDARD.LOOPCOUNTER*.

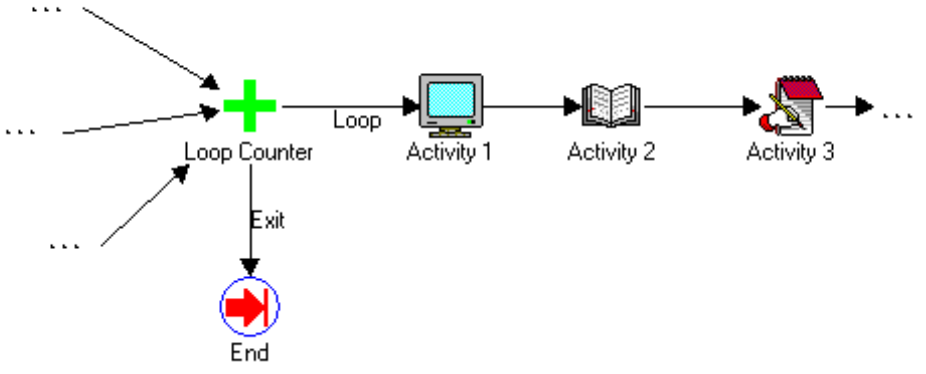
Activity Attribute

The Loop Counter activity has an activity attribute called Loop Limit. If the number of times that the Workflow Engine transitions to the Loop Counter activity is less than the

value specified in Loop Limit, the Loop Counter activity will complete with a result of Loop and the engine will take the 'Loop' transition to the next activity. If the number of times that the Workflow Engine transitions to the Loop Counter activity exceeds the value of Loop Limit, the activity will complete with a result of Exit and the engine will take the 'Exit' transition to an alternative activity.

For example, suppose your workflow process contains a branch of activities that can be transitioned to from multiple source activities, and you want to ensure that that particular branch of activities gets executed just once in the process. Include a Loop Counter activity as the first activity in that branch and specify the Loop Limit activity attribute value as `1`. Also draw an 'Exit' transition from the Loop Counter activity to an activity that you want the engine to execute if the Loop Counter activity is visited more than once, as shown in the diagram below.

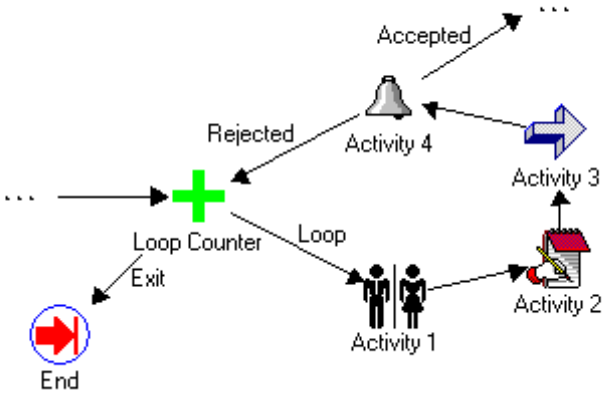
Example Loop Counter Activity in a Branch



Additionally, for example, as shown in the diagram below, you can include a Loop Counter activity as the initial activity in a loop. The value you specify for the Loop Limit activity attribute will designate the number of times the engine is allowed to traverse through the loop. If the number of visits to the Loop Counter activity exceeds the value set in Loop Limit, then the process moves along the 'Exit' transition to the designated activity.

In this example, the engine moves from the Loop Counter activity through Activities 1, 2, 3, and 4 in the loop. If the result of Activity 4 is 'Accepted,' the process moves along the 'Accepted' transition. Otherwise, if the result is 'Rejected,' the process moves along the 'Rejected' transition to return to the Loop Counter activity. If the item is rejected multiple times, once the number of visits to the Loop Counter activity exceeds the Loop Limit value, the process moves along the 'Exit' transition and ends.

Example Loop Counter Activity in a Loop



See: To Define Activity Attribute Values, page 4-13.

Start Activity

The Start activity marks the start of a process and does not perform any action. Although it is not necessary, you may include it in your process diagram to visually mark the start of a process as a separate node. This activity calls the PL/SQL procedure named *WF_STANDARD.NOOP*.

End Activity

The End activity marks the end of a process and does not perform any action. You can use it to return a result for a completed process by specifying a Result Type for the activity. Although it is not necessary, you may include it in your process diagram to visually mark the end of your process as a separate node. This activity calls the PL/SQL procedure named *WF_STANDARD.NOOP*.

Role Resolution Activity

The Role Resolution activity lets you identify a single user from a role comprised of multiple users. In a process diagram, place the Role Resolution activity in front of a notification activity and specify the performer of that notification activity to be a role consisting of several users. The Role Resolution activity selects a single user from that role and assigns the notification activity to that user.

This activity calls the PL/SQL procedure named *WF_STANDARD.ROLERESOLUTION*.

Activity Attributes

Use the Method activity attribute in the Role Resolution activity to specify how you want to resolve the role. A value of "Load Balance" compares how many open notifications from that activity each qualified user has and selects the user with the fewest open notifications from that activity. A value of "Sequential" selects a user from the role sequentially by determining the user that experienced the longest interval of time since last receiving a notification from that activity. See: To Define Activity Attribute Values, page 4-13.

Notify Activity

The Notify function activity lets you send a notification, where the message being sent is determined dynamically at runtime by a prior function activity. To use the Notify activity, you must model a prerequisite function activity into the process that selects one of several predefined messages for the Notify activity to send.

> **Important:** Since the Notify activity is locked against modifications at access level 0, you cannot change the result type from its value of `<None>`. Therefore, the message that the function activity dynamically selects must not have a result type, that is, it can only be an informative message that does not solicit a response.

> **Important:** If you want the Notify activity to send a message that requires a response, then you must copy and create your own version of the Notify activity. Since any one of several messages (with response attributes) can be sent by your version of the Notify activity, you must model into your process all the possible notification results that can be returned.

> **Note:** If you want to define an activity that always sends the same message, you should define a notification activity and not use this Notify function activity.

The Notify activity calls a PL/SQL procedure named *WF_STANDARD.NOTIFY*.

Activity Attributes

The Notify activity has three activity attributes:

- Message Name - the name of the predefined message to send. The prerequisite function activity that determines which message to send should store the name of that message in an item attribute. The Message Name activity attribute should reference that item attribute to determine the name of the message to send.
- Performer - the name of the role to which to send the notification message. If you load the roles from your database, you can select a constant role as the performer. Alternatively, you can set the performer to an item attribute that returns the name of a role at runtime.
- Expand Roles - takes as its value, the lookup codes `Yes` or `No`. Set Expand Roles to `Yes` if you wish to send an individual copy of the notification message to every user in the role.

See: To Define Activity Attribute Values, page 4-13.

Vote Yes/No Activity

The Vote Yes/No activity lets you send a notification to a group of users in a role and tally the Yes/No responses from those users. The results of the tally determine the activity that the process transitions to next.

The Vote Yes/No activity, classified as a notification activity, first sends a notification message to a group of users and then performs a PL/SQL post-notification function to tally the users' responses (votes).

Activity Attributes

The Vote Yes/No activity has three activity attributes:

- Percent Yes - The percentage of Yes votes cast in order for the activity to complete with a result of Yes.
- Percent No - The percentage of No votes cast in order for the activity to complete with a result of No

  Note: The values for the Percent Yes and Percent No attributes are both defined as null in order to use a Popularity voting method, in which the result is the response with the highest number of votes. See: Example Voting Methods, page 3-72.

- Voting Option - specify how the votes are tallied by selecting one of three values:
  - "Wait for All Votes" - the Workflow Engine waits until all votes are cast before tallying the results as a percentage of all the users notified. If a timeout condition occurs, the Workflow Engine calculates the resulting votes as a percentage of the total votes cast before the timeout occurred.
  - "Tally on Every Vote" - the Workflow Engine keeps a running tally of the cumulative responses as a percentage of all the users notified. If a timeout condition occurs, then the responses are tallied as a percentage of the total number of votes cast. Note that this option is meaningless if any of the custom response activity attributes have a blank value.
  - "Require All Votes" - the Workflow Engine evaluates the responses as a percentage of all users notified only after all votes are cast. If a timeout condition occurs, the Workflow Engine progresses along the standard timeout transition, or if none is available, raises an error, and does not tally any votes.

See: To Define Activity Attribute Values, page 4-13.

Related Topics

Voting Activity, page 3-69

Master/Detail Coordination Activities

The Master/Detail coordination activities let you coordinate the flow of master and detail processes. For example, a master process may spawn detail processes that need to be coordinated such that the master process continues only when every detail process has reached a certain point in its flow or vice versa.

When you spawn a detail process from a master process in Oracle Workflow, you are in effect creating a separate process with its own unique item type and item key. You define the master/detail relationship between the two processes by making a call to the Workflow Engine *SetItemParent* API after you call the *CreateProcess* API and before you call the *StartProcess* API when you create the detail process. See: SetItemParent, *Oracle Workflow API Reference*.

You can then use the two activities described below to coordinate the flow in the master and detail processes. One activity lets you pause a process and the other signals the halted process to continue. To use these activities, you place one activity in the master process and the other in each detail process.

Both activities contain two activity attributes that you use to identify the coordinating activity in the other process(es).

Wait for Flow Activity

Place this activity in a master or detail process to pause the flow until the other corresponding detail or master process completes a specified activity. This activity calls a PL/SQL procedure named *WF_STANDARD.WAITFORFLOW*.

You can use multiple Wait for Flow activities in the same workflow process. For example, if a master process launches two different sets of detail processes at different stages, the master process should include a separate Wait for Flow activity node for each set of detail processes. In this case, when you call the *WF_ENGINE.SetItemParent* API to associate a detail process with the master process, you must specify the parent context as the activity label name for the Wait for Flow activity node that corresponds to this detail process.

If the parent process contains only one Wait for Flow activity, however, you can leave the parent context parameter null when calling *WF_ENGINE.SetItemParent*.

Activity Attributes

The Wait for Flow activity contains two activity attributes:

- Continuation Flow - specify whether this activity is waiting for a corresponding "Master" or "Detail" process to complete.
- Continuation Activity - specify the label of the activity node that must complete in the corresponding process before the current process continues. The default value is `CONTINUEFLOW`.

See: To Define Activity Attribute Values, page 4-13.

Continue Flow Activity

Use this activity to mark the position in the corresponding detail or master process where, upon completion, you want the halted process to continue. This activity calls a PL/SQL procedure named *WF_STANDARD.CONTINUEFLOW*.

When a Continue Flow activity is executed, the *WF_STANDARD.CONTINUEFLOW* procedure checks whether the corresponding Wait for Flow activity is associated with any other processes that have not yet completed their Continue Flow activities. If so, the waiting process keeps waiting for those other processes. If the Wait for Flow activity is not waiting for any other processes, then the *WF_STANDARD.CONTINUEFLOW* procedure completes the Wait for Flow activity so that the process that was waiting now continues to the next activity.

Activity Attributes

The Continue Flow activity contains two activity attributes:

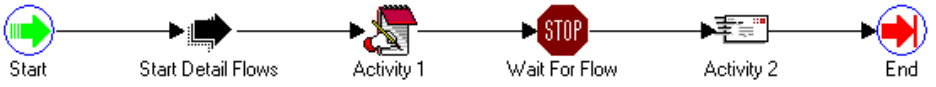
- Waiting Flow - specify whether the halted process that is waiting for this activity to complete is a "Master" or "Detail" flow.
- Waiting Activity - specify the label of the activity node in the halted process that is waiting for this activity to complete.

See: To Define Activity Attribute Values, page 4-13.

Example

The following figures show an example of how these coordination activities can be used. In the master process example, after the process begins with the Start activity, the Start Detail Flows activity initiates several detail processes. The master process then completes Activity 1 before it pauses at the Wait For Flow activity. Wait For Flow is defined to wait for all its detail processes to complete a Continue Flow activity before allowing the master process to transition to Activity 2 and finally end. An example of one of the detail processes below shows that after the detail process begins with the Start activity, it completes Activity A. When it reaches the Continue Flow activity, it signals to the Workflow Engine that the master process can now continue from the Wait For Flow activity. The detail process itself then transitions to Activity B and finally ends.

Example Master Process



Example Detail Process

Note: You can include a Wait for Flow activity in a master process without including a Continue Flow activity in one or more of its corresponding detail processes. However, in this case you must ensure that all detail processes without a Continue Flow activity are entirely completed before the Workflow Engine attempts to continue the master process based on the coordination activities that you do include. That is, at least one of the following must occur after the detail processes without a Continue Flow activity are completed:

- The master process reaches its Wait for Flow activity.
- A detail process that contains a Continue Flow activity completes that Continue Flow activity.

Otherwise, the Workflow Engine cannot properly coordinate the continuation of the master process.

If it does not matter when any of the detail processes complete before a master process continues (or when a master process completes before all the detail processes continue), then you simply omit both of the coordination activities from your master/detail processes.

Important: If you include a Continue Flow activity in a process, you must also include a Wait for Flow activity in its corresponding master

or detail process as defined by the activity attributes in the Continue Flow activity.

Assign Activity

The Assign activity lets you assign a value to an item attribute. This activity calls the PL/SQL procedure named *WF_STANDARD.ASSIGN*.

Activity Attributes

The Assign activity has an activity attribute called Item Attribute. Use Item Attribute to choose the item attribute that you want to assign a value to. Depending on the item attribute's format type, use the Date Value, Numeric Value, or Text Value activity attribute to specify the value that you want to assign to the item attribute.

Get Monitor URL Activity

The Get Monitor URL activity generates the URL for the Workflow Monitor diagram window and stores it in an item attribute that you specify. This activity calls the PL/SQL procedure named *WF_STANDARD.GETURL*.

Activity Attributes

The Get Monitor URL activity has two activity attributes:

- Item Attribute - choose the name of the item attribute that you want to use to store the URL for the Workflow Monitor window.
- Administration Mode - determine how the URL displays the Workflow Monitor window. If you set Administration Mode to "Yes", the URL displays the Workflow Monitor in 'ADMIN' mode, otherwise it displays the Workflow Monitor in 'USER' mode.

See: To Define Activity Attribute Values, page 4-13.

Get Event Property Activity

The Get Event Property activity lets you retrieve a property of an event message from the Business Event System and store the property value in an item attribute. This activity calls the PL/SQL procedure named *WF_STANDARD.GETEVENTPROPERTY*.

Activity Attributes

The Get Event Property activity has four activity attributes:

- Event - choose the item attribute of type event that contains the event message from which you want to retrieve a property.
- Property - the event property whose value you want to retrieve. This attribute takes as its value a lookup code from the Event Property lookup type. Possible values are "Priority," "Send Date," "Receive Date," "Correlation ID," "Event Parameter," "Event Name," "Event Key," "From Agent," "From Agent Name," "From Agent System," "To Agent," "To Agent Name," and "To Agent System." See: Event Message Structure, *Oracle Workflow API Reference*.
- Event Parameter - if you choose the Event Parameter property in the Property attribute, enter the name of the parameter whose value you want to retrieve. Oracle

Workflow uses this name to identify the parameter within the event message's parameter list. If you choose any property other than Event Parameter, leave this attribute blank.

- Item Attribute - the item attribute where you want to store the event property value.

See: To Define Activity Attribute Values, page 4-13.

Set Event Property Activity

The Set Event Property activity lets you set the value of a property in an event message from the Business Event System. This activity calls the PL/SQL procedure named *WF_STANDARD.SETEVENTPROPERTY*.

Activity Attributes

The Set Event Property activity has six activity attributes:

- Event - choose the item attribute of type event that contains the event message whose property you want to set.
- Property - the event property whose value you want to set. This attribute takes as its value a lookup code from the Event Property lookup type. Possible values are "Priority," "Send Date," "Receive Date," "Correlation ID," "Event Parameter," "Event Name," "Event Key," "From Agent," "From Agent Name," "From Agent System," "To Agent," "To Agent Name," and "To Agent System." See: Event Message Structure, *Oracle Workflow API Reference*.
- Event Parameter - if you choose the Event Parameter property in the Property attribute, enter the name of the parameter whose value you want to set. Oracle Workflow uses this name to identify the parameter within the event message's parameter list. If you choose any property other than Event Parameter, leave this attribute blank.
- Date Value - the value of type date that you want to set for the event property, if you choose the Send Date or Receive Date property.
- Numeric Value - the value of type number that you want to set for the event property, if you choose the Priority property.
- Text Value - the value of type text that you want to set for the event property, if you choose the Correlation ID, Event Parameter, Event Name, Event Key, From Agent Name, From Agent System, To Agent Name, or To Agent System property.

  Note: You must enter the value to set in the activity attribute that matches the data type of the event property you choose.

See: To Define Activity Attribute Values, page 4-13.

Compare Event Property Activity

The Compare Event Property activity lets you compare a property of an event message from the Business Event System with a test value that you specify. This activity calls the PL/SQL procedure named *WF_STANDARD.COMPAREEVENTPROPERTY*.

Activity Attributes

The Compare Event Property activity has six activity attributes:

- Event - choose the item attribute of type event that contains the event message whose property you want to compare to a test value.
- Property - the event property whose value you want to compare to a test value. This attribute takes as its value a lookup code from the Event Property lookup type. Possible values are "Priority," "Send Date," "Receive Date," "Correlation ID," "Event Parameter," "Event Name," "Event Key," "From Agent," "From Agent Name," "From Agent System," "To Agent," "To Agent Name," and "To Agent System." See: Event Message Structure, *Oracle Workflow API Reference*.
- Event Parameter - if you choose the Event Parameter property in the Property attribute, enter the name of the parameter whose value you want to compare to a test value. Oracle Workflow uses this name to identify the parameter within the event message's parameter list. If you choose any property other than Event Parameter, leave this attribute blank.
- Date Value - the test value of type date with which to compare the event property value, if you choose the Send Date or Receive Date property.
- Numeric Value - the test value of type number with which to compare the event property value, if you choose the Priority property.
- Text Value - the test value of type text with which to compare the event property value, if you choose the Correlation ID, Event Parameter, Event Name, Event Key, From Agent Name, From Agent System, To Agent Name, or To Agent System property.

The Compare Event Property activity uses the Comparison lookup type for a result code. Possible values are "Greater Than," "Less Than," "Equal," or "Null," if the test activity attribute value is null. You can guide your workflow process based on how the event property value compares to the test value. See: To Define Activity Attribute Values, page 4-13.

> **Note:** You must enter the test value in the activity attribute that matches the data type of the event property you choose. If you enter the test value in an inappropriate activity attribute, the Compare Event Property activity returns the "Null" result code.

XML Get Tag Value Activity

Use the XML Get Tag Value activity to retrieve data from the contents of an event message from the Business Event System. This functionality is currently only available for the standalone version of Oracle Workflow. This activity retrieves the data contained within a particular XML tag set in the event message and stores the data in an item attribute that you specify. The XML Get Tag Value activity calls the external Java function named *oracle.apps.fnd.wf.XMLGetTagValue*.

> **Note:** When the Workflow Engine encounters an external Java function activity, it places an entry on the 'Outbound' queue. To continue executing the activity, you must run the Java Function Activity Agent, which calls the appropriate Java function and places the result on the 'Inbound' queue. You must then run a background engine to process the 'Inbound' queue and complete the function activity. See: Setting Up the Java Function Activity Agent, *Oracle Workflow Administrator's Guide* and Setting Up Background Engines, *Oracle Workflow Administrator's Guide*.

Activity Attributes

The XML Get Tag Value activity has three activity attributes:

- Event - choose the item attribute of type event that contains the event message from which you want to retrieve data.
- Tag - the tag set within the event message from which you want to retrieve data. Specify the tag set in XPath notation. For example, for an XML document containing a purchase order, the XML path for the order number tag could be specified in the following format:

  ```
  /order/header/ordernumber
  ```

  The following example path locates the `ITEMNO` node on the third line of the purchase order document:

  ```
  /order/orderlines/line[3]/itemno
  ```

  The following example path locates the `COST` node on the second line of the purchase order document whose currency attribute is set to `"AUD"`. The notation `//` indicates that the specified node is located among the descendants of the root node.

  ```
  //line[2]/cost[@currency="AUD"]
  ```

  For more information, see the W3C Recommendation *XML Path Language (XPath)*.
- Item Attribute - choose the item attribute of type date, number, or text where you want to store the data. The type of the item attribute must match the type of the data that you want to retrieve.

See: To Define Activity Attribute Values, page 4-13.

XML Compare Tag Value Activities

Use the XML Compare Tag Value activities to compare data from an event message received through the Business Event System with a test value. This functionality is currently only available for the standalone version of Oracle Workflow. These activities compare the data contained within a particular XML tag set in the event message with the test value that you specify.

- **XML Compare Tag Value (Date)** - Use this activity to compare date values.
- **XML Compare Tag Value (Number)** - Use this activity to compare number values.
- **XML Compare Tag Value (Text)** - Use this activity to compare text values.

All the XML Compare Tag Value activities call the external Java function named *oracle.apps.fnd.wf.XMLCompareTag*.

> **Note:** When the Workflow Engine encounters an external Java function activity, it places an entry on the 'Outbound' queue. To continue executing the activity, you must run the Java Function Activity Agent, which calls the appropriate Java function and places the result on the 'Inbound' queue. You must then run a background engine to process the 'Inbound' queue and complete the function activity. See: Setting Up the Java Function Activity Agent, *Oracle Workflow Administrator's Guide* and Setting Up Background Engines, *Oracle Workflow Administrator's Guide*.

Activity Attributes

Each XML Compare Tag Value activity has three activity attributes:

- Event - choose the item attribute of type event that contains the event message whose data you want to compare.
- Tag - the tag set within the event message that contains the data you want to compare to a test value. Specify the tag set in XPath notation. For example, for an XML document containing an order, the XML path for the order number tag could be specified in the following format:

  ```
  /order/header/ordernumber
  ```

  The following example path locates the `ITEMNO` node on the third line of the purchase order document:

  ```
  /order/orderlines/line[3]/itemno
  ```

  The following example path locates the `COST` node on the second line of the purchase order document whose currency attribute is set to `"AUD"`. The notation `//` indicates that the specified node is located among the descendants of the root node.

  ```
  //line[2]/cost[@currency="AUD"]
  ```

  For more information, see the W3C Recommendation *XML Path Language (XPath)*.
- Value - the test value of type date, number, or text with which to compare the event message data.

The XML Compare Tag Value activities use the Comparison lookup type for a result code. Possible values are "Greater Than," "Less Than," "Equal," or "Null," if the test activity attribute value is null. You can guide your workflow process based on how the event message data compares to the test value. See: To Define Activity Attribute Values, page 4-13.

XML Transform Activity

The XML Transform activity lets you apply an XML style sheet to the payload of an event message from the Business Event System. This functionality is currently only available for the standalone version of Oracle Workflow. The resulting document is stored in an item attribute of type event. This activity calls the external Java function named *oracle.apps.fnd.wf.XSLTTransform*.

> **Note:** When the Workflow Engine encounters an external Java function activity, it places an entry on the 'Outbound' queue. To continue executing the activity, you must run the Java Function Activity Agent, which calls the appropriate Java function and places the result on the 'Inbound' queue. You must then run a background engine to process the 'Inbound' queue and complete the function activity. See: Setting Up the Java Function Activity Agent, *Oracle Workflow Administrator's Guide* and Setting Up Background Engines, *Oracle Workflow Administrator's Guide*.

Activity Attributes

The XML Transform activity has three activity attributes:

- Event - choose the item attribute of type event that contains the event message you want to transform.
- Stylesheet - a reference to the location of the style sheet that you want to apply. Specify this reference as a URL.
- New Document - choose the item attribute of type event where you want to store the new document produced by applying the style sheet.

See: To Define Activity Attribute Values, page 4-13.

Concurrent Manager Standard Activities

Oracle Applications provides some generic activities you can use to control your process if you are using the version of Oracle Workflow embedded in Oracle Applications. These activities are associated with the Concurrent Manager Functions item type but can be used within any process you define:

- Execute Concurrent Program Activity, page 5-17
- Submit Concurrent Program Activity, page 5-18
- Wait for Concurrent Program Activity, page 5-19

The Concurrent Manager Functions item type is automatically installed on your Oracle Applications workflow server. You can also access this item type from the file `fndwfaol.wft` located in the `$FND_TOP/admin/import` directory.

Execute Concurrent Program Activity

The Execute Concurrent Program activity is available only in the version of Oracle Workflow embedded in Oracle Applications. It submits an Oracle Applications concurrent program from your workflow process and waits for it to complete, at which point it updates the status of the activity and returns execution of the workflow process to the background engine. The concurrent program can complete with any of the following results, as defined by the Concurrent Program Status lookup type: `NORMAL`, `ERROR`, `WARNING`, `CANCELLED`, or `TERMINATED`. You should make sure all of these results are modelled into your process diagram.

> **Important:** To use the Execute Concurrent Program activity, you must ensure that the background engine is set up to run.

> **Important:** Generally, the context for your process' item type is always set if your session is initiated from an Oracle Applications form. However, if an interrupt occurs in your session, for example, due to a notification or blocking activity, you must ensure that the context is set by calling `FND_GLOBAL.APPS_INITIALIZE(user_id, resp_id, resp_appl_id)` in `SET_CTX` mode in your Selector/Callback function. See: Standard API for an Item Type Selector or Callback Function, page 6-10 and FNDSQF Routine APIs, *Oracle Applications Developer's Guide*.

The Execute Concurrent Program activity calls the standard Oracle Application Object Library API *FND_WF_STANDARD.EXECUTECONCPROGRAM*.

Activity Attributes

The Execute Concurrent Program activity has the following activity attributes:

- Application Short Name - Short name of the application to which the concurrent program is registered.
- Program Short Name - Short name of the concurrent program to run.
- Number of Arguments - Number of arguments required for the concurrent program.
- Item Attribute Name - Optional name of the item attribute to store the concurrent program request ID.
- Argument1, Argument2, ... Argument100 - Value of each concurrent program argument, ordered to match the correct syntax of the concurrent program. Up to 100 arguments are allowed, but you should only specify as many argument values as you define in the Number of Arguments activity attribute.

See: To Define Activity Attribute Values, page 4-13.

Submit Concurrent Program Activity

The Submit Concurrent Program activity is available only in the version of Oracle Workflow embedded in Oracle Applications. It submits an Oracle Applications concurrent program from your workflow process, but does not wait for it to execute or complete. Once this activity submits a concurrent request, the Workflow Engine continues with the next activity in the process.

> **Important:** Generally, the context for your process' item type is always set if your session is initiated from an Oracle Applications form. However, if an interrupt occurs in your session, for example, due to a notification or blocking activity, you must ensure that the context is set by calling `FND_GLOBAL.APPS_INITIALIZE(user_id, resp_id, resp_appl_id)` in `SET_CTX` mode in your Selector/Callback function. See: Standard API for an Item Type Selector or Callback Function, page 6-10.

The Submit Concurrent Program activity calls the standard Oracle Application Object Library API *FND_WF_STANDARD.SUBMITCONCPROGRAM*.

Activity Attributes

The Submit Concurrent Program activity has the following activity attributes:

- Application Short Name - Short name of the application to which the concurrent program is registered.
- Program Short Name - Short name of the concurrent program to run.
- Number of Arguments - Number of arguments required for the concurrent program.
- Item Attribute Name - Name of the item attribute to store the concurrent program request ID.
- Argument1, Argument2, ... Argument100 - Value of each concurrent program argument, ordered to match the correct syntax of the concurrent program. Up to 100 arguments are allowed, but you should only specify as many argument values as you define in the Number of Arguments activity attribute.

See: To Define Activity Attribute Values, page 4-13.

Wait for Concurrent Program Activity

The Wait for Concurrent Program activity is available only in the version of Oracle Workflow embedded in Oracle Applications. If you submit a concurrent program from your workflow process, you can use the Wait for Concurrent Program activity as a means of blocking the process from further execution until the concurrent program completes. When the concurrent program completes, this activity clears the block by updating the status of the activity and returning execution of the workflow process to the background engine. The concurrent program can complete with any of the following results, as defined by the Concurrent Program Status lookup type: `NORMAL`, `ERROR`, `WARNING`, `CANCELLED`, or `TERMINATED`. You should make sure all of these results are modelled into your process diagram.

> **Important:** To use the Wait for Concurrent Program activity, you must ensure that the background engine is set up to run.

The Wait for Concurrent Program activity calls the standard Oracle Application Object Library API *FND_WF_STANDARD.WAITFORCONCPROGRAM*.

Activity Attributes

The Wait for Concurrent Program activity has one activity attribute called Request ID, which should be set to the concurrent program request ID that you are waiting for to complete. See: To Define Activity Attribute Values, page 4-13.

6

Defining Procedures and Functions for Oracle Workflow

This chapter describes the standard APIs to use for Oracle Workflow PL/SQL and Java procedures and functions.

This chapter covers the following topics:

- Defining Procedures and Functions for Oracle Workflow
- Standard API for PL/SQL Procedures Called by Function Activities
- Standard API for Java Procedures Called by Function Activities
- Standard API for an Item Type Selector or Callback Function
- Standard APIs for "PL/SQL" Documents
- Standard API for an Event Data Generate Function
- Standard APIs for a Queue Handler
- Standard API for an Event Subscription Rule Function

Defining Procedures and Functions for Oracle Workflow

Oracle Workflow lets you integrate your own custom PL/SQL and Java procedures and functions at certain points in your workflow processes and in the Business Event System. To ensure that Oracle Workflow can properly execute your custom code, follow these standard APIs when developing your procedures and functions.

- Standard API for PL/SQL Procedures Called by Function Activities, page 6-2
- Standard API for Java Procedures Called by Function Activities, page 6-7
- Standard API for an Item Type Selector or Callback Function, page 6-10
- Standard APIs for "PL/SQL" Documents, page 6-13
- Standard API for an Event Data Generate Function, page 6-19
- Standard APIs for a Queue Handler, page 6-21
- Standard API for an Event Subscription Rule Function, page 6-23

Standard API for PL/SQL Procedures Called by Function Activities

All PL/SQL stored procedures that are called by function or notification activities in an Oracle Workflow process should follow this standard API format so that the Workflow Engine can properly execute the activity.

> **Important:** The Workflow Engine traps errors produced by function activities by setting a savepoint before each function activity. If an activity produces an unhandled exception, the engine performs a rollback to the savepoint, and sets the activity to the `ERROR` status. For this reason, you should never commit within the PL/SQL procedure of a function activity. The Workflow Engine never issues a commit as it is the responsibility of the calling application to commit.
>
> For environments such as database triggers or distributed transactions that do not allow savepoints, the Workflow Engine automatically traps "Savepoint not allowed" errors and defers the execution of the activity to the background engine.

The example in this section is numbered with the notation **(1)** for easy referencing. The numbers themselves are not part of the procedure.

```
(1)  procedure <procedure name>
       (itemtype in varchar2,
        itemkey in varchar2,
        actid in number,
        funcmode in varchar2,
        resultout out varchar2) is

(2)  <local declarations>

(3)  begin

     if ( funcmode = 'RUN' ) then
       <your RUN executable statements>
       resultout := 'COMPLETE:<result>';
       return;
     end if;

(4)  if ( funcmode = 'CANCEL' ) then
       <your CANCEL executable statements>
       resultout := 'COMPLETE';
       return;
     end if;

(5)  if ( funcmode = 'SKIP' ) then
       <your SKIP executable statements>
       resultout := 'COMPLETE:<result>';
       return;
     end if;

(6)  if ( funcmode = 'RETRY' ) then
       <your RETRY executable statements>
       resultout := 'COMPLETE:<result>';
       return;
     end if;
```

```
(7)  if ( funcmode = 'VALIDATE' ) then
       <your VALIDATE executable statements>
       resultout := 'COMPLETE';
       return;
     end if;

(8)  if ( funcmode = 'RESPOND' ) then
       <your RESPOND executable statements>
       resultout := 'COMPLETE';
       return;
     end if;

(9)  if ( funcmode = 'FORWARD' ) then
       <your FORWARD executable statements>
       resultout := 'COMPLETE';
       return;
     end if;

(10) if ( funcmode = 'TRANSFER' ) then
       <your TRANSFER executable statements>
       resultout := 'COMPLETE';
       return;
     end if;

(11) if ( funcmode = 'QUESTION' ) then
       <your QUESTION executable statements>
       resultout := 'COMPLETE';
       return;
     end if;

(12) if ( funcmode = 'ANSWER' ) then
       <your ANSWER executable statements>
       resultout := 'COMPLETE';
       return;
     end if;

(13) if ( funcmode = 'TIMEOUT' ) then
       <your TIMEOUT executable statements>
       if (<condition_ok_to_proceed>) then
         resultout := 'COMPLETE';
       else
         resultout := wf_engine.eng_timedout;
       end if;
       return;
     end if;

(14) if ( funcmode = '<other funcmode>' ) then
       resultout := ' ';
       return;
     end if;

(15) exception
       when others then
       WF_CORE.CONTEXT ('<package name>', '<procedure name>',
         <itemtype>, <itemkey>,
         to_char(<actid>), <funcmode>);
       raise;
```

```
(16) end <procedure name>;
```

(1) When the Workflow Engine calls a stored procedure for a function activity, it passes four parameters to the procedure and may expect a result when the procedure completes. The parameters are defined here:

itemtype
The internal name for the item type. Item types are defined in the Oracle Workflow Builder.

itemkey
A string that represents a primary key generated by the workflow-enabled application for the item type. The string uniquely identifies the item within an item type.

actid
The ID number of the activity from which this procedure is called.

funcmode
The execution mode of the activity. If the activity is a function activity, the mode can be 'RUN', 'CANCEL', 'SKIP', or 'RETRY'. If the activity is a notification activity, with a post-notification function, then the mode can be 'RESPOND','FORWARD','TRANSFER','QUESTION','ANSWER','VALIDATE','TIMEOUT', or 'RUN'. Other execution modes may be added in the future.

resultout
If a result type is specified in the Activities properties page for the activity in the Oracle Workflow Builder, this parameter represents the expected result that is returned when the procedure completes. The possible results are:

- COMPLETE:*<result_code>* - activity completes with the indicated result code. The result code must match one of the result codes specified in the result type of the function activity.
- WAITING - activity is pending, waiting on another activity to complete before it completes. An example is the Standard 'AND' activity.
- DEFERRED:*<date>* - activity is deferred to a background engine for execution until a given date. *<date>* must be of the format:

  ```
  to_char(<date_string>, wf_engine.date_format)
  ```

- NOTIFIED:*<notification_id>*:*<assigned_user>* - an external entity is notified that an action must be performed. A notification ID and an assigned user can optionally be returned with this result. Note that the external entity must call *CompleteActivity()* to inform the Workflow Engine when the action completes.
- ERROR:*<error_code>* - activity encounters an error and returns the indicated error code.

(2) This section declares any local arguments that are used within the procedure.

(3) The procedure body begins in this section with an IF statement. This section contains one or more executable statements that run if the value of funcmode is 'RUN'. One of the executable statements can return a result for the procedure. For example, a result can be 'COMPLETE:APPROVED'.

> **Note:** The Workflow Engine automatically runs a post-notification function in RUN mode after the Notification System completes execution of the post-notification function in RESPOND mode. The RUN mode

executable statements can perform processing such as vote tallying and determine what result to return for the notification activity.

(4) This section clears the activity and can contain executable statements that run if the value of `funcmode` is `'CANCEL'`. Often, this section contains no executable statements to simply return a null value, but this section also provides you with the chance to 'undo' something if necessary. An activity can have a `funcmode` of `'CANCEL'` in the special case where the activity is part of a loop that is being revisited.

The first activity in a loop must always have the Loop Reset option set in the activity properties Detail page. When the Workflow Engine encounters an activity that has already run, it verifies the activity's Loop Reset option. If this option is set to Reset, the engine then identifies the activities that belong in that loop and sets `funcmode` to `'CANCEL'` for those activities. Next, the engine transitions through the loop in forward order and executes each activity in `'CANCEL'` mode to clear all prior results for the activities so they can run again. See: Looping, *Oracle Workflow API Reference* and Loop Counter Activity, page 5-5.

(5) This section can contain executable statements that run if the value of `funcmode` is `'SKIP'`. For example, you can validate the result supplied for the skipped activity or notify an administrator that the activity was skipped. An activity can have a `funcmode` of `'SKIP'` if a user skips the activity from the Status Monitor, from an error notification, or using the *WF_ENGINE.HandleError* API.

> **Note:** If you want to prevent users from skipping the activity, return a result of `'#NOSKIP'` or `wf_engine.eng_noskip` instead of `'COMPLETE:<result>'`. When you set the result to specify that the activity cannot be skipped, Oracle Workflow raises an error instead of executing the skip operation. For example, set the result as follows:
>
> ```
> resultout := wf_engine.eng_noskip;
> ```

(6) This section can contain executable statements that run if the value of `funcmode` is `'RETRY'`. For example, you can verify that the retried activity is eligible to be reexecuted. An activity can have a `funcmode` of `'RETRY'` if a user retries the activity from the Status Monitor, from an error notification, or using the *WF_ENGINE.HandleError* API.

(7) This section is needed only for post-notification functions. Use this section to include execution statements that run if the value of `funcmode` is `'VALIDATE'`. This mode is executed before `'RESPOND'` mode, if you define this section in your post-notification function. For example, include execution statements that validate the response values before accepting and recording the response.

(8) This section is needed only for post-notification functions. Use this section to include execution statements that run if the value of `funcmode` is `'RESPOND'`, that is, when a `RESPOND` operation is performed. For example, include execution statements that validate the response to the notification. After the Notification System completes execution of the post-notification function in `RESPOND` mode, the Workflow Engine then runs the post-notification function again in `RUN` mode. See: Post-Notification Functions, *Oracle Workflow API Reference*.

(9) This section is needed only for post-notification functions. Use this section to include execution statements that run if the value of `funcmode` is `'FORWARD'`, that is, when a

notification's state changes to `'FORWARD'`. For example, include execution statements that validate the role to which the notification is being forwarded.

(10) This section is needed only for post-notification functions. Use this section to include execution statements that run if the value of `funcmode` is `'TRANSFER'`, that is, when a notification's state changes to `'TRANSFER'`. For example, include execution statements that validate the role to which the notification is being transferred.

(11) This section is needed only for post-notification functions. Use this section to include execution statements that run if the value of `funcmode` is `'QUESTION'`, that is, when a user requests more information about a notification from another user. For example, include execution statements that validate the role to which the request for more information is being sent.

(12) This section is needed only for post-notification functions. Use this section to include execution statements that run if the value of `funcmode` is `'ANSWER'`, that is, when a user answers a request. For example, include execution statements that validate the answering information.

> **Note:** For the `'VALIDATE'`, `'RESPOND'`, `'FORWARD'`, `'TRANSFER'`, `'QUESTION'`, and `'ANSWER'` modes, the `resultout` parameter is ignored, unless the returned value looks something like `'ERROR%'`. Therefore, if you do not want the Validate, Respond, Forward, Transfer, Question, or Answer operation to occur after having executed your post-notification function, you can do one of two things:
>
> - Return `'ERROR:<errcode>'` in the `resultout` parameter to convert it to a generic exception with the error code mentioned in the message.
> - Raise an exception directly in your procedure with a more informative error message. See: Post-Notification Functions, *Oracle Workflow API Reference* and Notification Model, *Oracle Workflow API Reference*.

(13) This section is needed only for post-notification functions. Use this section to include execution statements that run if a notification activity times out. You can include logic to test whether the workflow can proceed normally, and if so, to complete the activity so that the workflow can continue to the next activity. For example, if a Voting activity times out before all recipients respond, you can include logic that determines how to interpret the responses based on the current response pool and completes the activity with the appropriate result.

You should also include logic to return a result of `wf_engine.eng_timedout` if the workflow cannot proceed normally. Model any subsequent behavior in your process diagram using a <Timeout> transition to another activity. The Workflow Engine will follow the <Timeout> transition when the result `wf_engine.eng_timedout` is returned.

(14) This section handles execution modes other than those already specified. Other execution modes may be added in the future. Since your activity does not need to implement any of these other possible modes, it should simply return null.

(15) This section calls WF_CORE.CONTEXT() if an exception occurs, so that you can include context information in the error stack to help you locate the source of an error. See: CONTEXT, *Oracle Workflow API Reference*.

Standard API for Java Procedures Called by Function Activities

In standalone Oracle Workflow, you can create custom Java classes to be called by external Java function activities in an Oracle Workflow process. Java procedures that are called by function activities are implemented as classes that extend the WFFunctionAPI class. The custom Java classes should follow a standard API format so that they can be properly executed by the Oracle Workflow Java Function Activity Agent.

> **Note:** This functionality is currently only available for the standalone version of Oracle Workflow. It is not available in Oracle Applications.

> **Important:** The Workflow Engine traps errors produced by function activities by setting a savepoint before each function activity. If an activity produces an unhandled exception, the engine performs a rollback to the savepoint, and sets the activity to the `ERROR` status. For this reason, just as with PL/SQL procedures, you should never commit within the Java procedure of a function activity. The Workflow Engine never issues a commit as it is the responsibility of the calling application to commit.

Many Workflow Engine and Notification APIs have corresponding Java methods that your Java program can call to communicate with Oracle Workflow. The WFFunctionAPI and WFAttribute classes also contain methods that your Java program can call to access item type and activity attributes. See: Oracle Workflow Java Interface, *Oracle Workflow API Reference*, Workflow Function APIs, *Oracle Workflow API Reference*, and Workflow Attribute APIs, *Oracle Workflow API Reference*.

To invoke a custom Java class from within a workflow process, create an external Java function activity that calls the class. See: To Create a Function Activity, page 3-60.

Java function activities are implemented as external procedures. When the Workflow Engine reaches an external Java function activity, the Workflow Engine places a message on the Workflow 'Outbound' queue. The Java Function Activity Agent monitors this queue and calls the class indicated in the Function Name property for the function activity. When the Java procedure is complete, the Java Function Activity Agent enqueues the results onto the 'Inbound' queue. See: Setting Up the Java Function Activity Agent, *Oracle Workflow Administrator's Guide*.

> **Note:** These 'Outbound' and 'Inbound' queues are separate from the queues used for the Business Event System. See: Workflow Queue APIs, *Oracle Workflow API Reference*.

After a Java procedure completes, you must run a background engine to process the 'Inbound' queue and complete the function activity. Otherwise, the activity will remain in the `DEFERRED` status. See: Setting Up Background Engines, *Oracle Workflow Administrator's Guide*.

You must include the JAR files containing your custom classes in your `CLASSPATH` to make the classes accessible to the Java Function Activity Agent. The custom class files should reside on the same platform where the Java Function Activity Agent is run. The Java Function Activity Agent does not need to reside on the same tier as the database, however.

The example in this section is numbered with the notation **(1)** for easy referencing. The numbers themselves are not part of the procedure.

```
(1)  package oracle.apps.fnd.wf;

(2)  import java.io.*;
     import java.sql.*;
     import java.math.BigDecimal;
     import oracle.sql.*;
     import oracle.jdbc.driver.*;

     import oracle.apps.fnd.common.*;
     import oracle.apps.fnd.wf.engine.*;
     import oracle.apps.fnd.wf.*;

(3)  public class className extends WFFunctionAPI {

(4)    public boolean execute(WFContext pWCtx){

(5)      ErrorStack es = pWCtx.getWFErrorStack();
         try
         {

(6)        WFAttribute lAAttr = new WFAttribute();
           WFAttribute lIAttr = new WFAttribute();

(7)        loadActivityAttributes(pWCtx, itemType,
             itemKey, actID);
           loadItemAttributes(pWCtx);

(8)        lAAttr = getActivityAttr("AATTR");
           lIAttr = getItemAttr("IATTR");

(9)        <your executable statements>

(10)       lIAttr.value((Object)"NEWVALUE");
           setItemAttrValue(pWCtx, lIAttr);

(11)     }
         catch (Exception e)
         {
           es.addMessage("WF","WF_FN_ERROR");
           es.addToken("MODULE",this.getClass().getName());
           es.addToken("ITEMTYPE",itemType);
           es.addToken("ITEMKEY",itemKey);
           es.addToken("ACTID",actID.toString());
           es.addToken("FUNCMODE",funcMode);
           es.addToken("ERRMESSAGE",e.getMessage());
           return false;
         }

(12)     return true;
       }
     }
```

(1) By default, Java classes supplied by Oracle Workflow will be in the `oracle.apps.fnd.wf` package. This section is optional.

(2) For correct operation, you must include the listed packages.

(3) The custom Java class must extend the WFFunctionAPI class. This class provides class variables and methods that are essential to the operation of your function activity.

The parameters that are normally supplied to a PL/SQL function activity are available to the custom class as class variables. They are initialized prior to the call of the `boolean execute()` method. The `resultOut` and the `errorStack` are then passed back to the Oracle Workflow Engine.

The status of the completed activity will be set to `COMPLETE` unless a value is present in the `errorStack` variable. If there is a value in this variable, then the activity status will be set to `ERROR`. The contents of the `errorStack` variable can be set by using the ErrorStack class within the WFContext class. Refer also to sections 5 and 11 of this API for catching exceptions.

The predefined class variables include:

itemType
The internal name for the item type. Item types are defined in the Oracle Workflow Builder.

itemKey
A string that represents a primary key generated by the workflow-enabled application for the item type. The string uniquely identifies the item within an item type.

ActID
The ID number of the activity from which this procedure is called.

funcMode
The execution mode of the activity. Currently the only supported mode for external Java function activities is the `'RUN'` mode.

resultOut
If a result type is specified in the activity properties page for the activity in the Oracle Workflow Builder, this parameter represents the expected result that is returned when the procedure completes.

> **Note:** Unlike the `resultout` for a PL/SQL procedure called by a function activity, the `resultOut` for a Java procedure does not include a status code. In the Java API, only the result type value is required. The status of the activity will be set automatically by the Workflow Engine depending on whether there is a value in the `errorStack` variable.

(4) The custom Java class must implement the `boolean execute()` method. This will be the main entry point for your Java class. On successful completion, this method should return `true`.

(5) It is important to catch exceptions with your custom Java class and pass them back to the engine via the ErrorStack class. Refer also to section 11 of this API for catching exceptions.

(6) To access item and activity attributes, a WFAttribute class is provided.

(7) The values of the item attributes are not automatically available to the Java class. They are loaded on demand. The values can be loaded explicitly with the `void loadItemAttributes(WFContext)` or `void loadActivityAttributes(WFContext)` methods. The values are also loaded implicitly when you call the `WFAttribute getItemAttr(String)` or `WFAttribute getActivityAttr(String)` methods. This section is optional.

(8) The actual values of the item and activity attributes are accessed via the `WFAttribute getItemAttr(String)` and `WFAttribute getActivityAttr(String)` methods. If you have not explicitly loaded the values of the attributes, they will be automatically loaded at this point.

(9) This section contains your own executable statements. Usually, you add these executable statements after retrieving the required item and activity attribute details (section 8) and before setting item attribute values (section 10).

(10) Setting the value of an item attribute with the `void setItemAttrValue(WFContext, WFAttribute)` method writes the value of your local WFAttribute to the database. You need to set the values of the WFAttribute class with the `WFAttribute.value(Object)` method.

(11) It is important to catch exceptions within your custom Java class and pass them back to the engine via the ErrorStack class.

An unsuccessful execution of the external Java function activity will return `false`.

Note that any message in the WFContext.ErrorStack class variable will be passed back to the Workflow Engine and will cause the activity to be assigned a completion status of `ERROR`.

(12) A successfully executed external Java function activity will return `true`.

Standard API for an Item Type Selector or Callback Function

For any given item type, you can define a single function that operates as both a selector and a callback function. A selector function is a PL/SQL procedure that automatically identifies the specific process definition to execute when a workflow is initiated for a particular item type but no process name is provided. Oracle Workflow also supports using a callback function to reset or test item type context information. You can define one PL/SQL procedure that includes both selector and callback functionality by following a standard API.

Oracle Workflow can call the selector/callback function with the following commands:

- `RUN` - to select the appropriate process to start when either of the following two conditions occur:
  - A process is not explicitly passed to WF_ENGINE.CreateProcess.
  - A process is implicitly started by WF_ENGINE.CompleteActivity with no prior call to WF_ENGINE.CreateProcess.
- `SET_CTX` - to establish any context information for an item type and item key combination that a function activity in the item type needs in order to execute. The Workflow Engine calls the selector/callback function with this command each time it encounters a new item type and item key combination, to ensure that the correct context information is always set.
- `TEST_CTX` - to determine if the current item type context information is correct before executing a function. For example, the selector/callback function in `TEST_CTX` mode lets you check if a form can be launched with the current context information just before the Notification Details Web page launches a reference form. If the context is incorrect, the form cannot be launched and a message is displayed to that effect. See: To View the Details of a Notification, *Oracle Workflow User's Guide*.

The standard API for the selector/callback function is as follows. The example in this section is numbered with the notation **(1)** for easy referencing. The numbers themselves are not part of the procedure.

```
(1)  procedure <procedure name>
       (item_type in varchar2,
        item_key in varchar2,
        activity_id in number,
        command in varchar2,
        resultout in out varchar2) is

(2)  <local declarations>

(3)  begin

     if ( command = 'RUN' ) then
       <your RUN executable statements>
       resultout := '<Name of process to run>';
       return;
     end if;

(4)  if ( command = 'SET_CTX' ) then
       <your executable statements for establishing
         context information>
       return;
     end if;

(5)  if ( command = 'TEST_CTX' ) then
       <your executable statements for testing
         the validity of the current context information>
       resultout := '<TRUE or FALSE or NOTSET> ';
       return;
     end if;

(6)  if ( command = '<other command>' ) then
       resultout := ' ';
       return;
     end if;

(7)  exception
       when others then
       WF_CORE.CONTEXT ('<package name>', '<procedure name>',
         <itemtype>,<itemkey>,
         to_char(<actid>), <command>);
       raise;

(8)  end <procedure name>;
```

(1) When the Workflow Engine calls the selector/callback function, it passes four parameters to the procedure and may expect a result when the procedure completes. The parameters are defined here:

itemtype
The internal name for the item type. Item types are defined in the Oracle Workflow Builder.

itemkey
A string that represents a primary key generated by the workflow-enabled application for the item type. The string uniquely identifies the item within an item type.

actid
The ID number of the activity that this procedure is called from. Note that this parameter is always null if the procedure is called with the `'RUN'` command to execute the selector functionality.

command
The command that determines how to execute the selector/callback function. Either `'RUN'`, `'SET_CTX'`, or `'TEST_CTX'`. Other commands may be added in the future.

resultout
A result may be returned depending on the command that is used to call the selector/callback function.

If the function is called with `'RUN'`, the name of the process to run must be returned through the `resultout` parameter. If the function is called with `'SET_CTX'`, then no return value is expected. If the function is called with `'TEST_CTX'`, then the code must return `'TRUE'` if the context is correct, `'FALSE'` if the context is incorrect, or `'NOTSET'` if the context has not been initialized yet. If any other value is returned, Oracle Workflow assumes that this command is not implemented by the callback.

(2) This section declares any local arguments that are used within the procedure.

(3) The procedure body begins in this section with an `IF` statement. This section contains one or more executable statements that make up your selector function. It executes if the value of `command` is `'RUN'`. One of the executable statements should return a result for the procedure that reflects the process to run. For example, a result can be `'REQUISITION_APPROVAL'`, which is the name of an example process activity.

(4) This section contains one or more executable statements that set item type context information if the value of `command` is `'SET_CTX'`. The Workflow Engine calls the selector/callback function with this command each time it encounters a new item type and item key combination, before executing any function activities for that combination. This command is useful when you need to set item type context information in a database session before the activities in that session can execute as intended. For example, you might need to set up the responsibility and organization context for function activities that are sensitive to multi-organization data.

(5) This section contains one or more executable statements that validate item type context information if the value of `command` is `'TEST_CTX'`. The Workflow Engine calls the selector/callback function with this command to validate that the current database session context is acceptable before the Workflow Engine executes an activity. For example, this callback functionality executes just before the Notification Details Web page launches a reference form. The code in this section should return `'TRUE'` if the context is correct, `'FALSE'` if the context is incorrect, or `'NOTSET'` if the context has not been initialized yet.

- If the result is `'TRUE'`, the Workflow Engine keeps the current context.
- If the result is `'NOTSET'`, the Workflow Engine runs the function in `'SET_CTX'` mode to set the context.
- If the result is `'FALSE'` and the Workflow Engine permits context switching at this point in its processing, it runs the function in `'SET_CTX'` mode to set the correct context. However, if the result is `'FALSE'` but the Workflow Engine requires

the current context to be preserved, it defers the activity to be processed by a background engine instead.

(6) This section handles execution modes other than `'RUN'`, `'SET_CTX'` or `'TEST_CTX'` as others may be added in the future. Since your function does not need to implement any of these other possible commands, it should simply return null.

(7) This section calls WF_CORE.CONTEXT() if an exception occurs, so that you can include context information in the error stack to help you locate the source of an error. See: CONTEXT, *Oracle Workflow API Reference*.

Standard APIs for "PL/SQL" Documents

You can integrate a document into a workflow process by defining an attribute of type document for an item type, message, or activity. Oracle Workflow supports document types called "PL/SQL" documents, "PL/SQL CLOB" documents, and "PL/SQL BLOB" documents. A PL/SQL document represents data as a character string, a PL/SQL CLOB document represents data as a character large object (CLOB), and a PL/SQL BLOB document represents data as a binary large object (BLOB).

The document-type attribute that you create tells Oracle Workflow how to construct a dynamic call to a PL/SQL procedure that generates the document. You can embed a PL/SQL or PL/SQL CLOB document-type message attribute in a message body to display the document in a notification. You can also attach a PL/SQL, PL/SQL CLOB, or PL/SQL BLOB document-type message attribute to a message to include the document as an attachment to the notification.

The PL/SQL procedures that generate PL/SQL, PL/SQL CLOB, and PL/SQL BLOB documents must follow standard API formats.

> **Note:** If you create a PL/SQL document or a PL/SQL CLOB document that contains HTML, you should take security precautions to ensure that only the HTML code you intend to include is executed. If you retrieve any data from the database at runtime for inclusion in the document, use the *WF_NOTIFICATION.SubstituteSpecialChars()* API to substitute entity references for any HTML tag characters in that data, so that those characters will not be interpreted as HTML code and executed. See: SubstituteSpecialChars, *Oracle Workflow API Reference*.
>
> Note that you should not substitute entity references for HTML tags that you include in the document yourself. Otherwise, the document will not be displayed with your intended HTML formatting. You only need to perform this substitution for data that is retrieved from the database at runtime, which may be entered from an external source.

Related Topics

"PL/SQL" Documents, page 6-13

"PL/SQL CLOB" Documents, page 6-15

"PL/SQL BLOB" Documents, page 6-18

"PL/SQL" Documents

The PL/SQL procedure that generates a PL/SQL document must have the following standard API:

```
procedure <procedure name>
  (document_id in varchar2,
   display_type in varchar2,
   document in out varchar2,
   document_type in out varchar2)
```

The arguments for the procedure are as follows:

document_id
A string that uniquely identifies a document. This is the same string as the value that you specify in the default value field of the Attribute property page for a "PL/SQL" document (`plsql:<procedure>/<document_identifier>`). *<procedure>* should be replaced with the PL/SQL package and procedure name in the form of *package.procedure*. The phrase *<document_identifier>* should be replaced with the PL/SQL argument string that you want to pass directly to the procedure. The argument string should identify the document. For example: `plsql:po_wf.show_req/2034`. If you wish to generate the PL/SQL argument string value dynamically, create another item attribute, and reference that item attribute as "&*ITEM_ATTRIBUTE*" in place of the PL/SQL argument string. Then before any activity that references this other item attribute gets executed, call the WF_ENGINE.SetItemAttribute API to dynamically set the PL/SQL argument string value. For example: `plsql:po_wf.show_req/&POREQ_NUMBER`.

display_type
One of three values that represents the content type used for the notification presentation, also referred to as the requested type:

- `text/plain` - the document is embedded inside a plain text representation of the notification as viewed from an e-mail message. The entire e-mail message must be less than or equal to 32K, so depending on how large your e-mail template is, some of the plain text document that the procedure generates may get truncated. See: Modifying Your Message Templates, *Oracle Workflow Administrator's Guide*.
- `text/html` - the document is embedded inside an HTML representation of the notification as viewed from the Notification Web page, or the HTML attachment to an e-mail message. The procedure must generate an HTML representation of the document of up to 32K, but should not include top level HTML tags like `<HTML>` or `<BODY>` since the HTML page that the document is being inserted into already contains these tags. If you include top level HTML tags accidentally, Oracle Workflow removes the tags for you when the document attribute is referenced in a message body. Note that the procedure can alternatively generate a plain text document, as the notification system can automatically surround plain text with the appropriate HTML tags to preserve formatting.
- `' '` - the document is presented as a separate attachment to the notification. Any content type may be returned.

document
The outbound text buffer where up to 32K of document text is returned.

document_type
The outbound text buffer where the document content type is returned. Also referred to as the returned type. If no type is supplied, then `'text/plain'` is assumed.

Related Topics

To Define a Document Attribute, page 3-13

"PL/SQL CLOB" Documents

The PL/SQL procedure that generates a PL/SQL CLOB document must have the following standard API:

```
procedure <procedure name>
  (document_id in varchar2,
   display_type in varchar2,
   document in out clob,
   document_type in out varchar2)
```

A PL/SQL CLOB document that you include as an attachment to a notification can contain a PDF or RTF document or, if your database version is Oracle9*i* Database or higher, other binary data that is encoded to base64. You should first store the document in the database as a binary large object (BLOB) and then convert the document into a CLOB as part of the PL/SQL procedure that generates the CLOB. You can use the UTL_RAW.Cast_To_VARCHAR2 function to convert the data from the BLOB into VARCHAR2 data that you write to a CLOB. If your database version is Oracle9*i* Database or higher, you can optionally use the WF_MAIL_UTIL.EncodeBLOB procedure to encode the binary data to base64. See: UTL_RAW, *Oracle Supplied PL/SQL Packages and Types Reference* and EncodeBLOB, *Oracle Workflow API Reference*.

> **Note:** You can call WF_NOTIFICATION.WriteToClob() to help build a CLOB by appending a string of character data to it. See: WriteToClob, *Oracle Workflow API Reference*.

> **Note:** Oracle8*i* Database does not support base64 encoding, so if you are using Oracle8*i* Database, the WF_MAIL_UTIL.EncodeBLOB procedure is not available, and you cannot store binary data other than PDF or RTF documents in a PL/SQL CLOB document. This feature is available only if you are using Oracle9*i* Database or higher.
>
> However, both Oracle8*i* Database and Oracle9*i* Database and higher support the UTL_RAW.Cast_To_VARCHAR2 function, so you can store PDF and RTF documents in an attached PL/SQL CLOB document on any of these database versions.

The arguments for the procedure are as follows:

document_id

A string that uniquely identifies a document. This is the same string as the value that you specify in the default value field of the Attribute property page for a "PL/SQL CLOB" document (`plsqlclob:<procedure>/<document_identifier>`). *<procedure>* should be replaced with the PL/SQL package and procedure name in the form of *package.procedure*. The phrase *<document_identifier>* should be replaced with the PL/SQL argument string that you want to pass directly to the procedure. The argument string should identify the document. For example: `plsqlclob:po_wf.show_req_clob/2036`. If you wish to generate the PL/SQL argument string value dynamically, create another item attribute, and reference that item attribute as "&*ITEM_ATTRIBUTE*" in place of the PL/SQL argument string. Then before any activity that references this other item attribute gets executed, call the

WF_ENGINE.SetItemAttribute API to dynamically set the PL/SQL argument string value. For example: `plsqlclob:po_wf.show_req_clob/&POREQ_NUMBER`.

display_type

One of three values that represents the content type used for the notification presentation, also referred to as the requested type:

- `text/plain` - the document is embedded inside a plain text representation of the notification.
- `text/html` - the document is embedded inside an HTML representation of the notification as viewed from the Notification Web page. The procedure must generate an HTML representation of the document, but should not include top level HTML tags like `<HTML>` or `<BODY>` since the HTML page that the document is being inserted into already contains these tags. If you include top level HTML tags accidentally, Oracle Workflow removes the tags for you when the document attribute is referenced in a message body. Note that the procedure can alternatively generate a plain text document, as the notification system can automatically surround plain text with the appropriate HTML tags to preserve formatting.
- `' '` - the document is presented as a separate attachment to the notification. Any content type may be returned.

document

The outbound LOB locator pointing to where the document text is stored. This locator is a temporary LOB locator, so you must write your document text to this locator rather than replacing its value. If this value is overwritten, the temporary LOB is *not* implicitly freed. For more information, see: Temporary LOBs, *Oracle Application Developer's Guide - Large Objects (LOBs)*.

document_type

The outbound text buffer where the document content type is returned. Also referred to as the returned type. If no type is supplied, then `'text/plain'` is assumed. For a PDF or RTF document, this argument should specify an appropriate Multi-purpose Internet Mail Extension (MIME) type, such as `'application/pdf'` or `'application/rtf'`. You can also optionally specify a file name for the attachment as part of this argument. Use a semicolon (`';'`) to separate the file name from the preceding value in the argument, and specify the file name in the format `'name=<filename>'` with no spaces before or after the equal sign (`'='`). For example, you can set a value for the document type with the following command:

```
document_type := 'application/pdf; name=filename.pdf';
```

> **Note:** If your database version is Oracle9*i* Database or higher, and you are using the WF_MAIL_UTIL.EncodeBLOB API to encode binary data to base64 in order to store the data in this PL/SQL CLOB document, then the `document_type` parameter should specify an appropriate MIME type with a primary type of either `application` or `image`, such as `'application/doc'`, `'application/pdf'`, `'image/jpg'`, or `'image/gif'`. The MIME type must be followed by a semicolon (`';'`) and then by the encoding specification `'encoding=base64'` with no spaces before or after the equal sign. You can also optionally specify a file name for the attachment as part of this argument. Use a semicolon (`';'`) to separate the file name from the preceding value in the argument, and specify the file name in the format `'name=<filename>'`

with no spaces before or after the equal sign ('='). For example, you can set a value for the document type with the following command:

```
document_type := 'image/jpg; encoding=base64;
name=filename.jpg';
```

Example

The following example shows a sample procedure to produce a PL/SQL CLOB document that contains a PDF or RTF document, using the UTL_RAW.Cast_To_VARCHAR2 function to convert a BLOB to a CLOB.

```
procedure cdoc
  (document_id in varchar2,
   display_type in varchar2,
   document in out clob,
   document_type in out varchar2)

is

  bdoc blob;
  cdoc clob;
  content_type varchar2(100);

  lob_id number;
  amount number;

  bdoc_size number;
  block number := 10000;
  blockCount number;
  rawBuff RAW(32000);
  pos number;
  charBuff varchar2(32000);
  charBuff_size number;

  filename varchar2(200);
  location varchar2(200);
  data_type varchar2(100);

begin

  dbms_lob.createTemporary(cdoc, FALSE, dbms_lob.call);

  -- Determine the document to display from the
  -- Document ID
  lob_id := to_number(document_id);

  -- Obtain the BLOB version of the document
  select filename, location, content_type, data_type, bdata
  into filename, location, content_type, data_type, bdoc
  from sjm_lobs
  where id = lob_id;

  -- recast the BLOB to a CLOB
  bdoc_size := dbms_lob.getLength(bdoc);
  if block < bdoc_size then
    blockCount := round((bdoc_size/block)+0.5);
  else
```

```
    blockCount := 1;
  end if;
  pos := 1;
  for i in 1..blockCount loop
    dbms_lob.read(bdoc, block, pos, rawBuff);
    charBuff := utl_raw.cast_to_varchar2(rawBuff);
    charbuff_size := length(charBuff);
    dbms_lob.writeAppend(cdoc, charbuff_size, charBuff);
    pos := pos + block;
  end loop;

  -- Now copy the content to the document
  amount := dbms_lob.getLength(cdoc);
  dbms_lob.copy(document, cdoc, amount, 1, 1);

  -- Set the MIME type as a part of the document_type.
  document_type := content_type||'; name='||filename;

  exception
    when others then
      wf_core.context('LOBDOC_PKG', 'cdoc',
        document_id, display_type);
      raise;

end cdoc;
```

Related Topics

To Define a Document Attribute, page 3-13

"PL/SQL BLOB" Documents

The PL/SQL procedure that generates a PL/SQL BLOB document must have the following standard API:

```
procedure <procedure name>
  (document_id in varchar2,
   display_type in varchar2,
   document in out blob,
   document_type in out varchar2)
```

The arguments for the procedure are as follows:

document_id

A string that uniquely identifies a document. This is the same string as the value that you specify in the default value field of the Attribute property page for a "PL/SQL BLOB" document (`plsqlblob:<procedure>/<document_identifier>`). *<procedure>* should be replaced with the PL/SQL package and procedure name in the form of *package.procedure*. The phrase *<document_identifier>* should be replaced with the PL/SQL argument string that you want to pass directly to the procedure. The argument string should identify the document. For example: `plsqlblob:po_wf.show_req_blob/2038`. If you wish to generate the PL/SQL argument string value dynamically, create another item attribute, and reference that item attribute as "&*ITEM_ATTRIBUTE*" in place of the PL/SQL argument string. Then before any activity that references this other item attribute gets executed, call the

WF_ENGINE.SetItemAttribute API to dynamically set the PL/SQL argument string value. For example: `plsqlblob:po_wf.show_req_blob/&POREQ_NUMBER`.

display_type
For a PL/SQL BLOB document, this value should be `' '` to represent the content type used for the notification presentation, also referred to as the requested type:

`' '` - the document is presented as a separate attachment to the notification. Any content type may be returned.

document
The outbound LOB locator pointing to where the document text is stored. This locator is a temporary LOB locator, so you must write your document text to this locator rather than replacing its value. If this value is overwritten, the temporary LOB is *not* implicitly freed. For more information, see Temporary LOBs, *Oracle Application Developer's Guide - Large Objects (LOBs)*.

document_type
The outbound text buffer where the document content type is returned. Also referred to as the returned type. If no type is supplied, then `'text/plain'` is assumed. This argument should specify an appropriate MIME type with a primary type of either `application` or `image`, such as `'application/doc'`, `'application/pdf'`, `'image/jpg'`, or `'image/gif'`. You can also optionally specify a file name for the attachment as part of this argument. Use a semicolon (`';'`) to separate the file name from the preceding value in the argument, and specify the file name in the format `'name=<filename>'` with no spaces before or after the equal sign (`'='`). For example, you can set a value for the document_type with the following command:

```
document_type := 'image/jpg; name=filename.jpg';
```

Related Topics

To Define a Document Attribute, page 3-13

Standard API for an Event Data Generate Function

When you define an event in the Business Event System, you can assign the event a generate function that can produce the complete event data from the event name, event key, and an optional parameter list. The event data gives additional details to describe what occurred and can be structured as an XML document. You should specify a generate function if the application that raises the event will not produce the event data itself.

When an event is raised locally, the Event Manager checks each subscription before executing it to determine whether the subscription requires the event data. If the event data is required but is not already provided, the Event Manager calls the generate function for the event to produce the event data. The generate function returns the event data in character large object (CLOB) format.

> **Note:** If the event data is required but no generate function is defined for the event, Oracle Workflow creates a default set of event data using the event name and event key.

> **Note:** If the generate function is costly, and you want to return control to the calling application more quickly after raising the event, you can defer all the subscriptions that require the complete event data. Then

the Event Manager will not run the generate function until those subscriptions are executed at a later time. See: Deferred Subscription Processing, page 8-35.

Related Topics

To Define an Event, page 8-5

To Define an Event Subscription, page 8-38

Parameter List Structure, *Oracle Workflow API Reference*

Standard API for a PL/SQL Generate Function

A PL/SQL generate function must have the following standard API:

```
function <function_name>
  (p_event_name in varchar2,
   p_event_key in varchar2
   p_parameter_list in wf_parameter_list_t default null)
  return clob;
```

The arguments for the function are as follows:

p_event_name
The internal name of the event.

p_event_key
A string generated when the event occurs within a program or application. The event key uniquely identifies a specific instance of the event.

p_parameter_list
An optional list of additional parameter name and value pairs for the event.

Standard API for a Java Generate Function

In Oracle Applications only, you can assign an event a Java generate function to be executed in the middle tier, instead of a PL/SQL generate function. A Java generate function must be a Java class using the following Java interface:

```
public interface GenerateInterface
{
  public String generate(BusinessEvent event,
      WorkflowContext context)
    throws BusinessEventException;
}
```

The arguments for the API are as follows:

event
The BusinessEvent object. The payload passed to the BusinessEvent class must be serializable.

context
Workflow context information, including the Log object which can be used for logging.

For more information, see the Oracle Workflow Javadoc.

Standard APIs for a Queue Handler

When you define an agent in the Business Event System, you must assign the agent a queue handler. The queue handler is a package that translates between the standard Workflow event message format defined by the `WF_EVENT_T` datatype and the message format required by the queue associated with the agent. In Oracle Applications, a Java queue handler translates between the BusinessEvent object format and the queue's message format.

Oracle Workflow provides two standard queue handlers for queues that use the `WF_EVENT_T` format, WF_EVENT_QH for normal processing and WF_ERROR_QH for error queues. Oracle Workflow also provides a standard queue handler named WF_EVENT_OJMSTEXT_QH for queues that use JMS Text messages as their payload format.

You can also create your own custom queue handlers for queues that use other formats. If you create a custom queue handler, you must provide standard enqueue and dequeue APIs in your package. Java queue handlers must also include some other standard APIs.

Related Topics

Event Message Structure, *Oracle Workflow API Reference*

Agents (for standalone Oracle Workflow), page 8-17

Agents (for Oracle Applications), page 8-76

Mapping Between WF_EVENT_T and SYS.AQ$_JMS_TEXT_MESSAGE, *Oracle Workflow API Reference*

Standard APIs for a PL/SQL Queue Handler

Enqueue

The Enqueue procedure in a queue handler package must enqueue an event message onto a queue associated with an outbound agent. You can optionally specify an override agent where you want to enqueue the event message. Otherwise, the event message is enqueued on the From Agent specified within the message. The Enqueue procedure transforms the event message's header information if necessary to enqueue the message in the format required by the queue.

When an event message is being sent, the generic WF_EVENT.Enqueue procedure determines which queue handler is associated with the specified outbound agent and calls the Enqueue procedure in that queue handler to enqueue the message.

The PL/SQL Enqueue procedure must have the following standard API:

```
procedure enqueue
  (p_event in WF_EVENT_T,
   p_out_agent_override in WF_AGENT_T);
```

The arguments for the procedure are as follows:

p_event
The event message.

p_out_agent_override
The outbound agent on whose queue the event message should be enqueued.

Dequeue

The Dequeue procedure in a queue handler package must dequeue an event message from the queue associated with the specified inbound agent, selecting the message to dequeue by the message priority. The procedure transforms the event message's header information if necessary and returns the event message in the standard `WF_EVENT_T` structure. Additionally, the Dequeue procedure can set the date and time when the message is dequeued into the `RECEIVE_DATE` attribute of the event message.

When an event message is being received, the WF_EVENT.Listen procedure determines which queue handler to use with the specified inbound agent and calls the Dequeue procedure in that queue handler to dequeue the message.

The PL/SQL Dequeue procedure must have the following standard API:

```
procedure dequeue
  (p_agent_guid in raw,
   p_event out WF_EVENT_T);
```

The arguments for the procedure are as follows:

p_agent_guid
The globally unique identifier of the inbound agent from whose queue the event message should be dequeued.

p_event
The event message.

Standard APIs for a Java Queue Handler

In Oracle Applications only, you can provide a Java queue handler to be executed during Java event message processing in the middle tier, instead of a PL/SQL queue handler. A Java queue handler must be a Java class using the following Java interface:

```
public interface QueueHandlerInterface
{
  public void init(Connection conn,
                   AgentEO agent,
                   Log log,
                   String uniqueLogId,
                   Properties props)
    throws QueueHandlerException;

  public String enqueue(BusinessEvent event)
    throws QueueHandlerException;

  public BusinessEvent dequeue()
    throws QueueHandlerException;

  public void destroy();

  public String getMsgId()
    throws QueueHandlerException;

  public CLOB getEventData();
}
```

In addition to the enqueue and dequeue APIs, a Java queue handler must also contain methods to initialize the connection to the agent, destroy objects created during queue

processing after the processing is complete, retrieve a message ID from a message on the queue, and retrieve the CLOB reference to the event data of the last business event dequeued by the queue handler.

The arguments for the init method are as follows:

conn
The JDBC connection used to establish the connection with the queue.

agent
The AgentEO object that contains the information for this agent and its queue.

log
The Log object which can be used for logging.

uniqueLogId
The log ID for recording debug messages.

props
The property mapping for enqueue and dequeue operations.

The argument for the enqueue method is as follows:

event
The BusinessEvent object to be enqueued.

For more information, see the Oracle Workflow Javadoc.

Standard API for an Event Subscription Rule Function

When you define an event subscription, you choose a function called a rule function to run on the event message. Oracle Workflow provides a standard default rule function named WF_RULE.Default_Rule to perform basic subscription processing. The default rule function includes the following actions:

- Sending the event message to a workflow process, if specified in the subscription definition
- Sending the event message to an agent, if specified in the subscription definition

See: Default_Rule, *Oracle Workflow API Reference*.

Oracle Workflow also provides additional standard rule functions that you can use for advanced processing, testing, debugging, or other purposes. Commonly used rule functions including the following:

- Default_Rule2 - Performs the basic subscription processing only if the parameter list within the event message includes parameters whose names and values match all the parameters defined for the subscription. See: Default_Rule2, *Oracle Workflow API Reference*.
- Default_Rule3 - Sets the parameter name and value pairs from the subscription parameters into the parameter list within the event message before performing the basic subscription processing. See: Default_Rule3, *Oracle Workflow API Reference*.
- SendNotification - Sends a notification as specified by the parameter list within the event message. This rule function lets you send a notification outside of a workflow process. See: SendNotification, *Oracle Workflow API Reference*.

- Instance_Default_Rule - Sends the event to all existing workflow processes that have eligible receive event activities waiting to receive it, identified by a business key attribute. See: Instance_Default_Rule, *Oracle Workflow API Reference*.

See: Event Subscription Rule APIs, *Oracle Workflow API Reference*.

You can extend your subscription processing by creating custom rule functions. Custom rule functions must be defined according to a standard API.

A rule function may read from or write to the event message or perform any other database action. However, you should never commit within a rule function. The Event Manager never issues a commit as it is the responsibility of the calling application to commit. Additionally, the rule function must not change the connection context in any way, including security and NLS settings.

> **Note:** If your rule function writes to the event message, any subsequent subscriptions executed on the event will access the changed message.

If the subscription processing that you want to perform for an event includes several successive steps, you may find it advantageous to define multiple subscriptions to the event with simple rule functions that you can reuse, rather than creating complex specialized rule functions that cannot be reused. You can use the phase numbers for the subscriptions to specify the order in which they should be executed.

By default, the Event Manager uses the event key as the correlation ID for the event message when no other correlation ID is specified. If you want to specify a different correlation ID, you can use *WF_EVENT_FUNCTIONS_PKG.AddCorrelation* to add a correlation ID to the event message, either by calling this function within your custom rule function or by defining another subscription that uses *WF_EVENT_FUNCTIONS_PKG.AddCorrelation* as its rule function. See: AddCorrelation, *Oracle Workflow API Reference*.

If you want to send the event message to a workflow process or to an agent after running custom code on the message, you must either include the send processing in your rule function, or define a separate subscription that uses the default rule function to perform the send processing.

- Call *WF_ENGINE.Event()* to send the event message to a workflow process.
- Call *WF_EVENT.Send()* to send the event message to an agent.
- Call *WF_RULE.Default_Rule()* to include the default subscription processing that can send the event message both to a workflow process and to an agent.

> **Note:** When you define a subscription in the Event Manager, you can define the workflow item type, workflow process name, out agent, to agent, priority, and parameters for your send processing, as well as defining the rule function. Any rule function can access these send attributes, but if you do not use the default rule function, you must explicitly include the send processing in your custom rule function if you want to send the event from the same subscription.

Related Topics

To Define an Event Subscription, page 8-38

Event Message Structure, *Oracle Workflow API Reference*

Workflow Core APIs, *Oracle Workflow API Reference*

Event Subscription Rule APIs, *Oracle Workflow API Reference*

Event(), *Oracle Workflow API Reference*

Send(), *Oracle Workflow API Reference*

Default_Rule, *Oracle Workflow API Reference*

SetErrorInfo(), *Oracle Workflow API Reference*

Standard API for a PL/SQL Subscription Rule Function

The standard API for a PL/SQL rule function is as follows. This section is numbered with the notation **(1)** for easy referencing. The numbers themselves are not part of the procedure.

```
(1)  function <function_name>
       (p_subscription_guid in raw,
        p_event in out WF_EVENT_T)
       return varchar2 is

(2)  <local declarations>

(3)  begin
       <your executable statements>

(4)  <optional code for WARNING>
       WF_CORE.CONTEXT('<package name>',
                       '<function name>',
                       p_event.getEventName( ),
                       p_subscription_guid);
       WF_EVENT.setErrorInfo(p_event, 'WARNING');
       return 'WARNING';

(5)  return 'SUCCESS';

(6)  exception
       when others then
         WF_CORE.CONTEXT('<package name>',
                         '<function name>',
                         p_event.getEventName( ),
                         p_subscription_guid);
         WF_EVENT.setErrorInfo(p_event, 'ERROR');
         return 'ERROR';

(7)  end;
```

(1) When the Event Manager calls the rule function, it passes two parameters to the function and expects a return code when the function completes. The parameters are defined here:

p_subscription_guid
The globally unique identifier for the subscription.

p_event
The event message.

The function must return one of the following status codes:

- `SUCCESS` - The rule function completed successfully.
- `WARNING` - A warning condition occurred. The rule function reports a warning message using the Workflow Core error APIs and sets the warning information into the event message. The Event Manager places a copy of the event message on the WF_ERROR queue, but subscription processing continues.
- `ERROR` - An error occurred. The rule function reports an error message using the Workflow Core error APIs and sets the error information into the event message. The Event Manager halts subscription processing for this event, rolls back any subscriptions already executed for the event, and places the event message on the WF_ERROR queue.

(2) This section declares any local arguments that are used within the function.

(3) The procedure body begins in this section with one or more executable statements that make up your rule function.

(4) This optional section calls WF_CORE.CONTEXT() if a warning condition occurs, so that you can include context information in the error stack to help you locate the source of an error. It also sets the warning information into the event message and returns the status code `'WARNING'`. See: CONTEXT, *Oracle Workflow API Reference*.

(5) This section returns the status code `'SUCCESS'` when the rule function's normal processing completes successfully.

(6) This section calls WF_CORE.CONTEXT() if an exception occurs, so that you can include context information in the error stack to help you locate the source of an error. It also sets the error information into the event message and returns the status code `'ERROR'`. See: CONTEXT, *Oracle Workflow API Reference*.

> **Note:** If you raise an exception in the rule function, the Event Manager rolls back all subscription processing for the event and raises the error to the calling application. In this case the event message is not placed on the WF_ERROR queue.

Standard API for a Java Subscription Rule Function

In Oracle Applications only, you can provide a Java subscription rule function to be executed in the middle tier, instead of a PL/SQL rule function. A Java rule function must be a Java class using the following Java interface:

```
public interface SubscriptionInterface
{
  void onBusinessEvent(Subscription eo, BusinessEvent event,
                       WorkflowContext ctx)
    throws BusinessEventException;
}
```

The arguments for the API are as follows:

eo
The Subscription object, which provides information about the subscription such as the phase number and any parameters.

event
The BusinessEvent object, which provides information about the business event that occurred, including the event name, event key, event data, and the optional payload object.

ctx
Workflow context information, including the database connection and the Log object which can be used for logging.

If the execution of the rule function does not succeed, the subscription can pass the error details to the Event Manager by encapsulating the details in a BusinessEventException.

Ensure that your rule function Java class is part of the `JAVA_TOP` that is included in the `AF_CLASSPATH` on the concurrent processing tier for your Oracle Applications installation.

> **Note:** When raising an event from Java, you can also set a serializable object as a payload for the event. The payload is available to all Java subscriptions to the event.

For more information, see the Oracle Workflow Javadoc.

7

Testing a Workflow Definition

This chapter tells you how to test your workflow definitions using the standalone Launch Processes Web page or Oracle Applications Developer Studio.

This chapter covers the following topics:

- Testing Workflow Definitions
- Testing Workflow Definitions Using the Developer Studio

Testing Workflow Definitions

The standalone version of Oracle Workflow provides a Web-based interface called Launch Processes for you to test any workflow definition you define and save to the database. Launch Processes is accessible only to users belonging to the Workflow Administrator role.

> **Note:** For information about the Developer Studio testing pages in Oracle Applications, see: Testing Workflow Definitions Using the Developer Studio, page 7-2.

Although you can run the Launch Processes Web page against any Oracle Workflow database, we advise that you create a separate environment for testing purposes. To test a workflow definition, you should set up the following in your test environment:

- Define test users and roles. You can test against the users and roles predefined in the Oracle Workflow demonstration data model. See: Installing the Requisition Data Model, page 10-4.
- If you plan to use the Notifications Web page to view notifications, you need to define your test users and roles in your Web security system. Refer to your Web server documentation for more information.
- If you plan to use e-mail to view notifications, you can send all notifications to a single test e-mail address by setting the Test Address configuration parameter when you configure a notification mailer. For more information, please refer to the Oracle Workflow Manager sections of the Oracle Enterprise Manager online help for standalone Oracle Workflow.

To Test a Workflow Definition:

1. Use a Web browser to connect to the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.
2. Select the Launch Processes link to display the Launch Processes Web page.

Important: Note that you can also connect to this page directly using the secured URL:

`<webagent>/wf_initiate.ItemType`

`<webagent>` represents the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

Important: This is a secured page, so if you have not yet logged on as a valid workflow administrator in the current Web session, you will be prompted to do so before the page appears.

3. The Launch Processes page displays all the item type definitions stored in the database except the Oracle Workflow seeded item types: `Wferror`, `Wfmail`, and `Wfstd`. The internal name and description for each item type also appears. Select the item type that owns the workflow process definition you wish to test.
4. Use the Initiate Workflow Web page to specify the details for the process you wish to launch. To initiate an instance of a workflow process, you need to specify:
   - A unique item key for the process instance.

     Note: The item key for a process instance can only contain single-byte characters. It cannot contain a multibyte value.

   - A user-defined key that you want to use to identify the process.
   - The name of the process to test.
   - An optional process owner.
   - Values for any item type attributes associated with the item type of the process.

   Select OK. To initiate the workflow process, the Initiate Workflow Web page calls the Workflow Engine *CreateProcess* and *StartProcess* APIs for the item type and item key specified. It also calls the Workflow Engine *SetItemOwner* and *SetItemAttr* APIs to set the process owner and all the item type attributes to the values specified.
5. The Workflow Monitor Activities List for your initiated process instance appears. The Activities List displays the status of the activities that have been executed. You can also select the View Diagram button to display the status of the process graphically in the Workflow Monitor. See: To Filter Activities in the Activities List, *Oracle Workflow Administrator's Guide*.
6. If the process you are testing contains notifications, you can navigate back to the Workflow Home page and select the Find Notifications link to find the outstanding Notifications that require responses to complete the process. Alternatively, if you prefer to test the notification responses via e-mail, you can connect to the e-mail test account you specified for the notification mailer to respond to the outstanding notifications for your process.

Testing Workflow Definitions Using the Developer Studio

The Developer Studio in Oracle Applications lets you view workflow definitions stored in your database. If you have workflow administrator privileges, you can also run a test

workflow process for a workflow definition. Workflow administrator privileges are assigned in the Workflow Configuration page. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

Although you can run a test workflow process against any Oracle Workflow database, it is recommended that you create a separate environment for testing purposes. You should set up test users and roles in your test environment before you begin.

> **Note:** If you plan to use e-mail to view notifications, you can send all notifications to a single test e-mail address by setting the Test Address configuration parameter when you configure a notification mailer. For more information, please refer to the Oracle Workflow Manager sections of the Oracle Applications Manager online help.

> **Note:** For information about the Launch Processes page in standalone Oracle Workflow, see: Testing Workflow Definitions, page 7-1.

To Search for Workflow Definitions in the Developer Studio:

1. Use a Web browser to navigate to the Developer Studio, using a responsibility and navigation path specified by your system administrator. See: Oracle Workflow Developer Navigation Paths, page A-1.
2. Search for the workflow definitions you want to review. The search criteria are:
   - Workflow Type - Select the workflow item type you want to review. The display name for the workflow type you select populates the Workflow Type field, and the internal name for the workflow type you select populates the Type Internal Name field.
   - Type Internal Name - Enter the internal name of the workflow type you want to view, if you want to enter the internal name directly instead of selecting a value. You can enter a partial value to search for workflow types whose internal names begin with that value.
3. If you have workflow administrator privileges, you can run a test workflow process by selecting the run icon for that workflow definition. See: To Run a Test Workflow Process, page 7-3.

   > **Note:** If a workflow definition does not include any runnable processes, its run icon is disabled.
   >
   > Additionally, if a workflow process begins with a Receive event activity, you cannot use the Developer Studio to test the process. Instead, you should raise a test event from the Event Manager to trigger an event subscription that launches the process. See: To Raise a Test Event, page 8-57.

To Run a Test Workflow Process:

1. Navigate to the Run Workflow page.
2. In the Workflow Identifier region, specify the identifying details for the workflow process you want to run.
   - Workflow Owner - Optionally specify an owner for the process. The default value is your user name.

- Item Key - A unique item key to identify the process instance.

  Note: The item key for a process instance can only contain single-byte characters. It cannot contain a multibyte value.

- User Key - An optional user-friendly identifier that you can use to locate the workflow process in the Status Monitor and other Oracle Workflow user interface components.
- Process - The name of the workflow process to test. If the workflow item type includes only one runnable process, Oracle Workflow automatically displays that process. If the workflow item type includes more than one runnable process, select the process that you want to test.

3. The Workflow Attributes region displays any item attributes associated with the workflow item type. Enter values for any item attributes that are required to initiate the process.
4. Select the Run Workflow button. To initiate the workflow process, the Run Workflow page calls the Workflow Engine *CreateProcess()* and *StartProcess()* APIs for the specified workflow item type and item key. It also calls the Workflow Engine *SetItemOwner()* and *SetItemAttr()* APIs to set the process owner and all the item type attributes to the specified values.
5. To view the status of the workflow process, choose the Status Monitor tab and search for the process in the Workflows search page. See: Accessing the Administrator Monitor, *Oracle Workflow Administrator's Guide*.
6. If the process you are testing includes notifications, choose the Notifications tab to view and respond to the notifications in the Worklist. See: To View Notifications from the Advanced Worklist, *Oracle Workflow User's Guide*.

   Alternatively, if you have set the notification preference of the recipients to receive e-mail notifications, you can use an e-mail reader to view and respond to the notifications. If you specified a test e-mail address for the notification mailer, you can connect to that e-mail account to access all notifications for the process. See: Reviewing Notifications via Electronic Mail, *Oracle Workflow User's Guide*.

8

Managing Business Events

This chapter tells you how to manage business events using the Oracle Workflow Event Manager Web pages.

This chapter covers the following topics:

- Managing Business Events
- Event Manager for Standalone Oracle Workflow
- Events
- Defining Events
- Systems
- Defining Systems
- Agents
- Defining Agents
- Event Subscriptions
- Defining Event Subscriptions
- Raising Events
- Signing Up Systems
- Synchronizing Systems
- Event Manager for Oracle Applications
- Events
- Defining Events
- Event Subscriptions
- Defining Event Subscriptions
- Agents
- Defining Agents
- Systems
- Defining Systems
- Workflow Agent Ping/Acknowledge

Managing Business Events

The Oracle Workflow Business Event System is an application service that leverages the Oracle Advanced Queuing (AQ) infrastructure to communicate business events between systems. The Business Event System consists of the Event Manager and workflow process event activities.

The Event Manager contains a registry of business events, systems, named communication agents within those systems, and subscriptions indicating that an event is significant to a particular system. Events can be raised locally or received from an external system or the local system through AQ. When a local event occurs, the subscribing code is executed in the same transaction as the code that raised the event, unless the subscriptions are deferred.

Subscriptions can include the following types of processing:

- Executing custom code on the event information
- Sending event information to a workflow process
- Sending event information to other queues or systems

Business events are represented within workflow processes by event activities. By including event activities in a workflow process, you can model complex processing or routing logic for business events beyond the options of directly running a predefined function or sending the event to a predefined recipient. See: Event Activity, page 3-54.

The uses of the Business Event System include:

- **System integration messaging hubs** - Oracle Workflow with the Business Event System can serve as a messaging hub for complex system integration scenarios. The Event Manager can be used to "hard-wire" routing between systems based on event and originator. Workflow process event activities can be used to model more advanced routing, content-based routing, transformations, error handling, and so on.
- **Distributed applications messaging** - Applications can supply generate and receive event message handlers for their business entities. For example, message handlers can be used to implement master/copy replication for distributed applications.
- **Message-based system integration** - You can set up subscriptions which cause messages to be sent from one system to another when business events occur. In this way, you can use the Event Manager to implement point-to-point messaging integration.
- **Business event-based workflow processes** - You can develop sophisticated workflow processes that include advanced routing or processing based on the content of business events.
- **Non-invasive customization of packaged applications** - Analysts can register interesting business events for their internet or intranet applications. Users of those applications can register subscriptions to those events to trigger custom code or workflow processes.

Event Manager for Standalone Oracle Workflow

The Oracle Workflow Event Manager lets you register interesting business events that may occur in your applications, the systems among which events will be communicated, named communication agents within those systems, and subscriptions indicating that an event is significant to a particular system. You can use the Event

Manager Web pages to define and maintain these events, systems, agents, and subscriptions.

> **Note:** You must have workflow administrator privileges to access the Event Manager Web pages.

> **Note:** When defining internal names for Business Event System objects in the Event Manager, use only characters from the ASCII character set.

You can use the Workflow XML Loader to upload and download XML definitions for Business Event System objects between a database and a flat file. See: Using the Workflow XML Loader, *Oracle Workflow Administrator's Guide*.

When an event is raised by a local application or received from a local or external system, the Event Manager executes any subscriptions to that event. Depending on the action defined in the subscription, the Event Manager may call custom code, send the event information to a workflow process, or send the event information to an agent.

After you finish setting up the Business Event System, you can use the Event Manager to raise events manually, sign up systems to receive business events from each other, and synchronize systems with each other. You can test your setup using Workflow Agent Ping/Acknowledge.

> **Note:** For information about the version of the Event Manager available in Oracle Applications, see: Event Manager for Oracle Applications, page 8-49.

You can use the Oracle Workflow Manager component available through Oracle Enterprise Manager to review the statuses of the messages being held on local queues. For more information, please refer to the Oracle Workflow Manager online help.

Related Topics

Events, page 8-3

Systems, page 8-12

Agents, page 8-17

Event Subscriptions, page 8-29

Raising Events, page 8-44

Signing Up Systems, page 8-45

Synchronizing Systems, page 8-47

Workflow Agent Ping/Acknowledge, page 8-99

Events

A business event is an occurrence in an internet or intranet application or program that might be significant to other objects in a system or to external agents. For instance, the creation of a purchase order is an example of a business event in a purchasing application. You can define your significant events in the Event Manager.

Oracle Workflow provides several predefined events for significant occurrences within the Business Event System. See: Predefined Workflow Events, page 9-1

When an event occurs in an application on your local system, an event key must be assigned to uniquely identify that particular instance of the event. Then the event must be raised to the Event Manager.

You can raise an event by any of the following methods:

- Raise the event from the application where the event occurs using the *WF_EVENT.Raise()* API. See: Raise, *Oracle Workflow API Reference*.
- Raise the event from a workflow process using a Raise event activity. See: Event Activity, page 3-62.
- Raise the event manually using the Raise Events page. See: Raising Events, page 8-44.

Additionally, the Event Manager can receive events sent from the local system or remote systems.

When you define an event in the Event Manager, you must assign it a unique internal name, which is case-sensitive. The suggested format for these internal names is a compound structure of identifiers separated by periods (`.`) as follows:

```
<company>.<family>.<product>.<component>.<object>.<event>
```

This format allows you to organize the events you define into a classification hierarchy.

You can also define event groups that let you associate any events you want with each other and reference them as a group in event subscriptions. An event group is a type of event composed of a set of individual member events. The internal names of event groups should follow the same format as the names of individual events. Once you have defined an event group, you can register a subscription to the group rather than having to create separate subscriptions for each individual event within it. The subscription will be executed whenever any one of the group's member events occurs.

> **Note:** Event groups cannot be used to raise events. You must raise each event individually.

Any detail information needed to describe what occurred in an event, in addition to the event name and event key, is called the event data. For example, the event data for a purchase order event includes the item numbers, descriptions, and cost. The event data can be structured as an XML document.

The application where the event occurs can include the event data when raising the event to the Event Manager. If the application will not provide the event data, you should specify a generate function for the event that can produce the complete event data from the event name, event key, and an optional parameter list. The generate function must follow a standard API. See: Raise, *Oracle Workflow API Reference* and Standard API for an Event Data Generate Function, page 6-19.

The Event Manager checks each subscription before executing it to determine whether the subscription requires the event data. If the event data is required but is not already provided, the Event Manager calls the generate function for the event to produce the event data. If the event data is required but no generate function is defined for the event, Oracle Workflow creates a default set of event data using the event name and event key.

> **Note:** If the generate function is costly, and you want to return control to the calling application more quickly after raising the event, you can defer all the subscriptions that require the complete event data. Then the Event Manager will not run the generate function until those subscriptions are executed at a later time. See: Deferred Subscription Processing, page 8-35.

If you use a program to create event definitions automatically, the program can set its own name and brief identifier as the owner name and owner tag for the events. The program can then use this identifying information to locate the events that it owns. You can use the Edit Event and Edit Group pages to update the owner name and owner tag manually if necessary.

Each event is assigned a customization level that determines whether you can update the event definition. Oracle Workflow uses the customization level to protect Oracle seed data and to preserve your customizations in an upgrade. An event can have one of the following customization levels:

- Core - No changes can be made to the event definition. This level is used only for events seeded by Oracle.
- Limit - The event status can be updated to Enabled or Disabled, but no other changes can be made to the event definition. This level is used only for events seeded by Oracle.
- User - Any property in the event definition can be updated. This level is automatically set for events that you define.

See: Access Protection for Business Event System Data, *Oracle Workflow Administrator's Guide*.

Related Topics

To Define an Event, page 8-5

To Define an Event Group, page 8-7

To Find Events, page 8-10

To Update or Delete an Event, page 8-11

Defining Events

To Define an Event:

1. Use a Web browser to connect to the following URL:

   `<webagent>/wf_event_html.listevents`

   Replace `<webagent>` with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

   > **Important:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current Web session, you will be prompted to do so before the page appears. You must have

workflow administrator privileges to access the Event Manager Web pages.

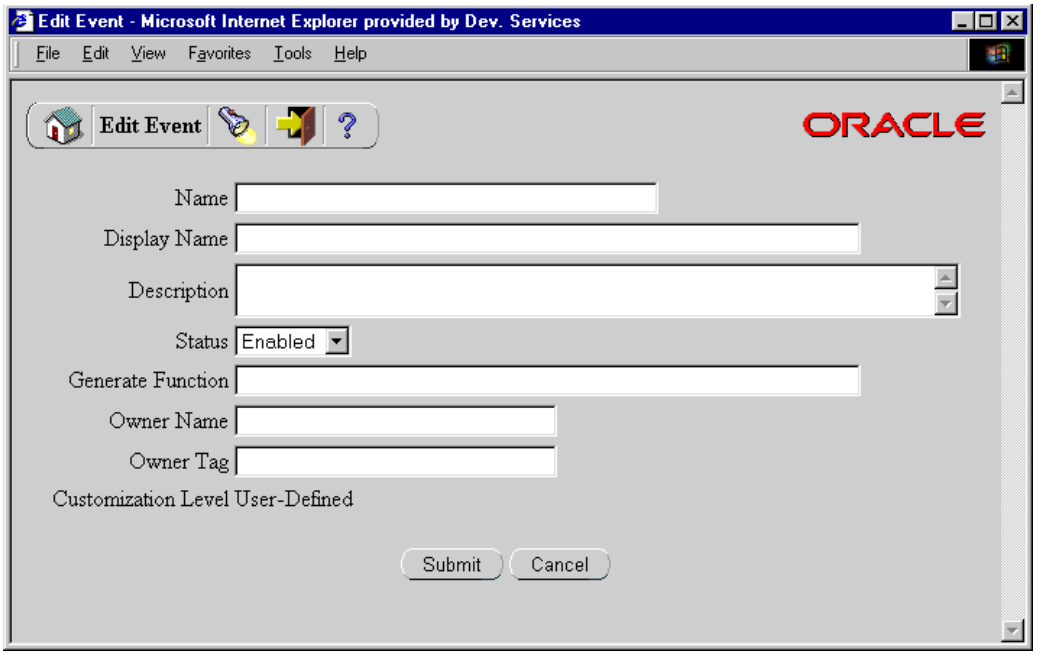
Note: You can also access the Events Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

2. The Events page appears, displaying a list of existing events. The Events page summarizes the internal name, display name, type, and status of each event.

   Choose the Add Event button to open the Edit Event page.

   Note: For seeded events with a customization level of Limit, you can only update the event status. For seeded events with a customization level of Core, you cannot update any properties; you can only view the event definition.

Edit Event Page



3. Enter the internal name of the event in the Name field. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying an event. The internal name is case-sensitive. The suggested format is a compound structure of identifiers separated by periods (`.`) as follows:

   `<company>.<family>.<product>.<component>.<object>.<event>`

4. Enter a Display Name for the event. This name appears in the Events list.
5. Enter an optional description for the event.
6. In the Status field, select Enabled or Disabled as the event status. If you disable an event, it still remains in the Events list for reference, but you cannot use the event in active subscriptions.

7. If you are defining an event that occurs on your local system, enter the Generate Function for the event. The generate function is a PL/SQL procedure that can produce the complete event data from the event name, event key, and an optional parameter list. See: Standard API for an Event Data Generate Function, page 6-19.
8. You can optionally identify the program or application that owns the event by entering the program name in the Owner Name field and the program ID in the Owner Tag field. The Owner Name and Owner Tag are not required if you are defining an event manually in the Edit Event page. However, if you use a program to create event definitions automatically, the Event Manager displays the owner information set by that program in these fields. You can use the Edit Event page to update this information manually if necessary.
9. You can review the customization level for the event.
   - Core - No changes can be made to the event definition. This level is used only for events seeded by Oracle.
   - Limit - The event status can be updated to Enabled or Disabled, but no other changes can be made to the event definition. This level is used only for events seeded by Oracle.
   - User - Any property in the event definition can be updated. This level is automatically set for events that you define.
10. Choose the Submit button to save the event and return to the Events page. The Events page displays an updated list of events.

    You can also choose the Cancel button to return to the Events page without saving the event.

To Define an Event Group:

1. Use a Web browser to connect to the following URL:

   `<webagent>/wf_event_html.listevents`

   Replace `<webagent>` with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

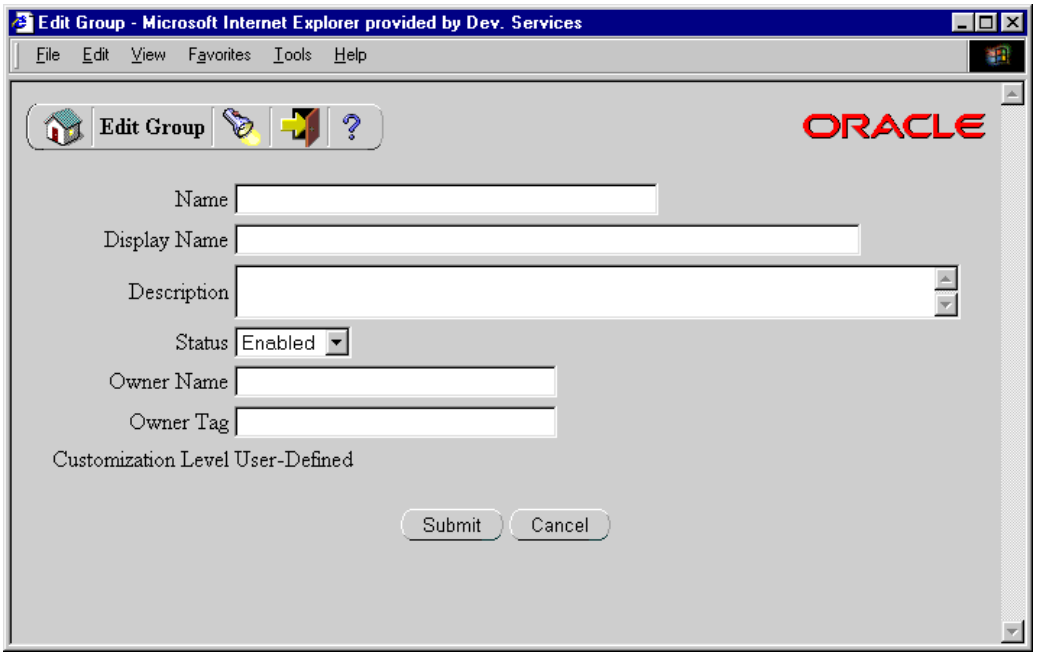
   > **Important:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current Web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager Web pages.

   > **Note:** You can also access the Events Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

2. The Events page appears, displaying a list of existing events. Choose the Add Group button to open the Edit Group page.

   > **Note:** For seeded event groups with a customization level of Limit, you can only update the event group status. For seeded events with a customization level of Core, you cannot update any properties; you can only view the event group definition.

Edit Group Page



3. Enter the internal name of the event group in the Name field. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying an event group. The internal name is case-sensitive and should have a compound structure of identifiers separated by periods (.) in the following format:

   <company>.<family>.<product>.<component>.<object>.<event>

4. Enter a Display Name for the event group. This name appears in the Events list.
5. Enter an optional description for the event group.
6. In the Status field, select Enabled or Disabled as the event group status. If you disable an event group, it still remains in the Events list for reference, but you cannot use the event group in active subscriptions.
7. You can optionally identify the program or application that owns the event group by entering the program name in the Owner Name field and the program ID in the Owner Tag field. The Owner Name and Owner Tag are not required if you are defining an event group manually in the Edit Group page. However, if you use a program to create event group definitions automatically, the Event Manager displays the owner information set by that program in these fields. You can use the Edit Group page to update this information manually if necessary.
8. You can review the customization level for the event group.
   - Core - No changes can be made to the event group definition. This level is used only for event groups seeded by Oracle.
   - Limit - The event group status can be updated to Enabled or Disabled, but no other changes can be made to the event group definition. This level is used only for event groups seeded by Oracle.

- User - Any property in the event group definition can be updated. This level is automatically set for event groups that you define.

9. Choose the Submit button to save the event group.

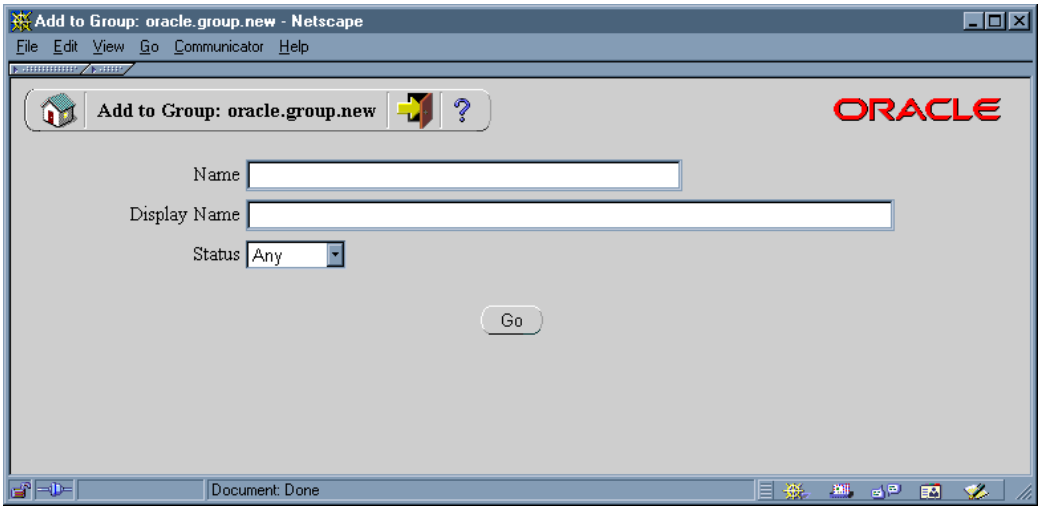
   Note: You can also choose the Cancel button to return to the Events page without saving the event group.

   After you save the event group definition, the Edit Group page displays the list of member events for that group, including the name, display name, and status of each event.

10. To add a member event to the group, choose the Add Event button.

    Note: An event group can contain only individual events as its members. It cannot contain another group.

Add to Group Page



11. In the Add to Group page that appears, enter search criteria to locate the event you want to add. The search criteria are:
    - Name - enter the internal name of the event you want to add.
    - Display Name - enter the display name of the event you want to add.
    - Status - choose Enabled or Disabled as the status of the event you want to add. Choose Any to search for events of any status.
12. Choose the Go button. The Add to Group page displays a list of events that match your search criteria.
13. Select the event or events that you want to add to your event group. You can choose the Select All button to select all the events in the list, or choose the Deselect All button to deselect all the events in the list.

    If you want to search for different events, enter new search criteria and choose the Go button. The Add to Group page displays the list of events that match your new search criteria.

You can also choose the Cancel button to cancel your current search and return to your previous search results.

14. When you have finished selecting the events you want to add, choose the Add button to add the selected events to your event group. The Edit Group page appears, displaying the updated list of event group members.
15. Choose the Submit button to save the event group definition.

    Note: You can also choose the Cancel button to return to the Add to Group page with your latest search results.

16. If you want to remove a member event from the group, select the event or events you want to delete in the Edit Group page. Choose Select All to select all the events in the list, or choose Deselect All to deselect all the events in the list.
17. Choose the Delete button to remove the selected events from your event group. The Edit Group page displays the updated list of event group members.

    Note: Removing an individual member event from an event group does not delete the event definition for the individual event. The individual event remains in the Events list.

18. To view the subscriptions that reference an event, choose the schedule icon in the Subs column for that event. The Event Subscriptions page appears, displaying the list of subscriptions to the event.

    Note: For events that do not have any subscriptions yet, a blank schedule icon appears. For events that do have subscriptions referencing them, a full schedule icon appears.

    You can begin defining a new subscription to the event by choosing the Add Subscription button in the Event Subscriptions page. The Edit Subscription page appears with the event name automatically entered in the Event Filter field. See Defining Event Subscriptions, page 8-29.

19. To update an event, choose the pencil icon in the Edit column for that event. The Edit Event page appears. Make your changes to the event definition and save your work. See: To Define an Event, page 8-5.

To Find Events:

1. Use a Web browser to connect to the following URL:

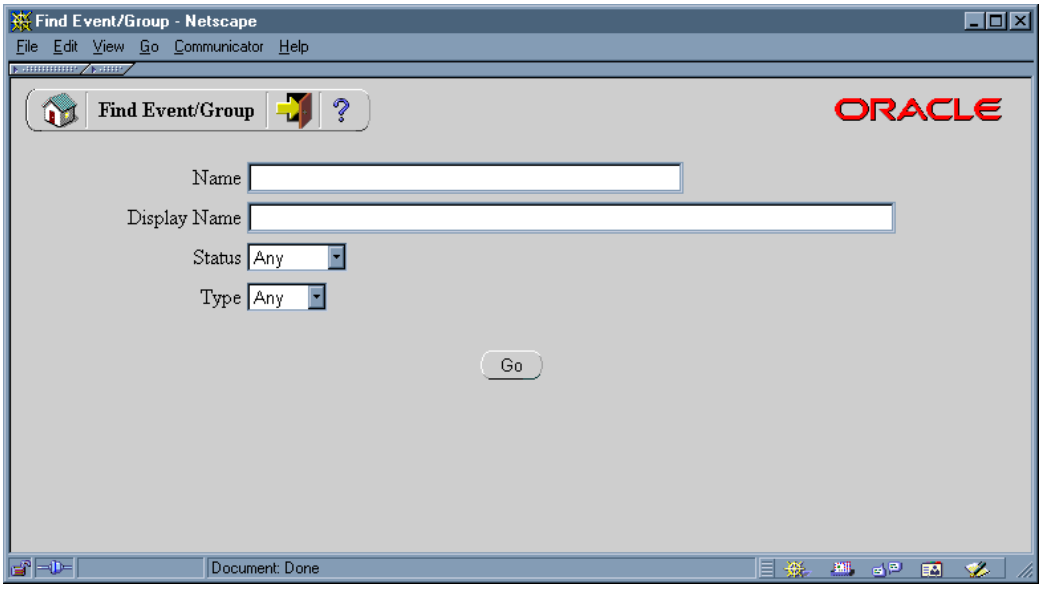
   `<webagent>/wf_event_html.findevent`

   Replace `<webagent>` with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

   Important: This URL accesses a secured page, so if you have not yet logged on as valid user in the current Web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager Web pages.

Note: You can also access the Find Event/Group Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

Find Event/Group Page



2. The Find Event/Group page appears. The Find Event/Group page lets you enter search criteria to locate specific events. The search criteria are:
   - Name - enter the internal name of the event you want to display.
   - Display Name - enter the display name of the event you want to display.
   - Status - choose Enabled or Disabled as the status of the events you want to display, or choose Any to display events of any status.
   - Type - choose Event or Group as the type of the events you want to display, or choose Any to display events of any type.
3. Choose the Go button. The Events page appears, displaying a list of events that match your search criteria.

To Update or Delete an Event:

1. Locate the event you want in the Events page. You can use the Find Event/Group page to find the event that you want and display the event in the Events page. See: To Find Events, page 8-10.
2. To view the subscriptions that reference an event, choose the schedule icon in the Subs column for that event. The Event Subscriptions page appears, displaying the list of subscriptions to the event.

   Note: For events that do not have any subscriptions yet, a blank schedule icon appears. For events that do have subscriptions referencing them, a full schedule icon appears.

You can begin defining a new subscription on the event by choosing the Add Subscription button in the Event Subscriptions page. The Edit Subscription page appears with the event name automatically entered in the Event Filter field. See Defining Event Subscriptions, page 8-29.

3. To update an event, choose the pencil icon in the Edit column for that event. The Edit Event page appears. Make your changes to the event definition and save your work. See: To Define an Event, page 8-5.

   Note: For seeded events with a customization level of Limit, you can only update the event status. For seeded events with a customization level of Core, you cannot update any properties; you can only view the event definition.

4. To delete an event, choose the trash icon in the Delete column for that event, and choose OK in the confirmation window that appears. You can also choose Cancel in the confirmation window to return to the Events page without deleting the event.

   Note: You can only delete events that do not have any subscriptions referencing them and that do not belong to any event groups.

   Additionally, you cannot delete any event seeded by Oracle that has a customization level of Core or Limit.

Systems

A system is a logically isolated software environment such as a host machine or database instance. You should define each system to or from which you will communicate events in the Event Manager.

When you define a system, you can specify whether it is associated with a master system from which you want it to receive Event Manager object definition updates.

Each system can expose one or more addressable points of communication, called agents. After you define your systems, you should define the agents within those systems that you will use to communicate business events. See: Agents, page 8-17.

Local System

When you install Oracle Workflow in a database, that database is automatically defined as a system in the Event Manager and set as the local system in the Global Workflow Preferences page. The following table lists the default properties of the local system definition.

Local System Properties

System Property	Value
Name	*<database global name>*
Display Name	*<database global name>*
Description	Local System Created by Oracle Workflow Configuration Assistant
Master	(blank)

You can update the local system definition if necessary.

Oracle Workflow sets the status of the local system to Enabled by default. After you finish setting up the Business Event System, you can use the Global Workflow Preferences page to set the system status that you want for event processing. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

Related Topics

To Define a System, page 8-13

To Find Systems, page 8-15

To Update or Delete a System, page 8-16

Defining Systems

To Define a System:

1. Use a Web browser to connect to the following URL:

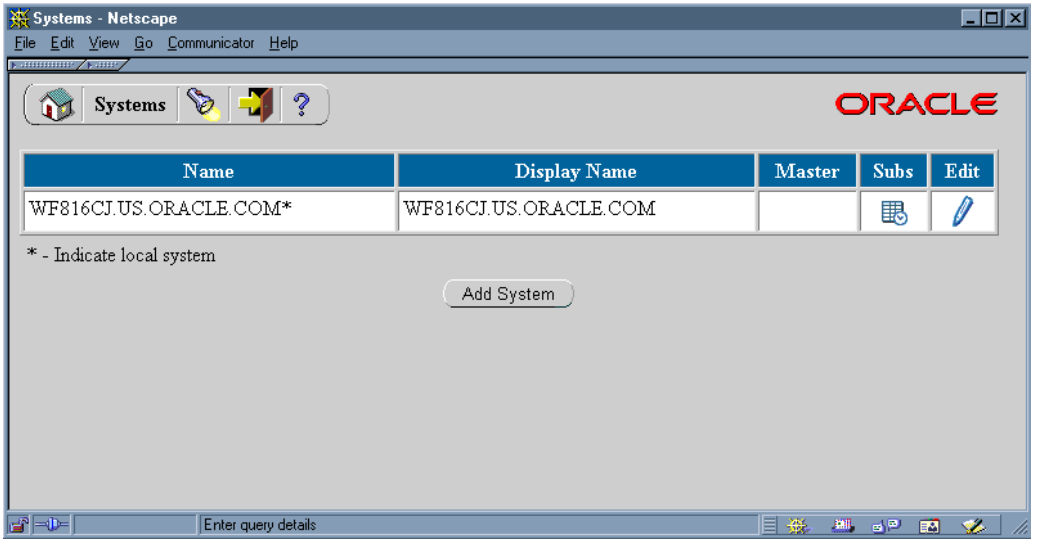
   `<webagent>/wf_event_html.listsystems`

   Replace `<webagent>` with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

   Important: This URL accesses a secured page, so if you have not yet logged on as valid user in the current Web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager Web pages.

   Note: You can also access the Systems Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

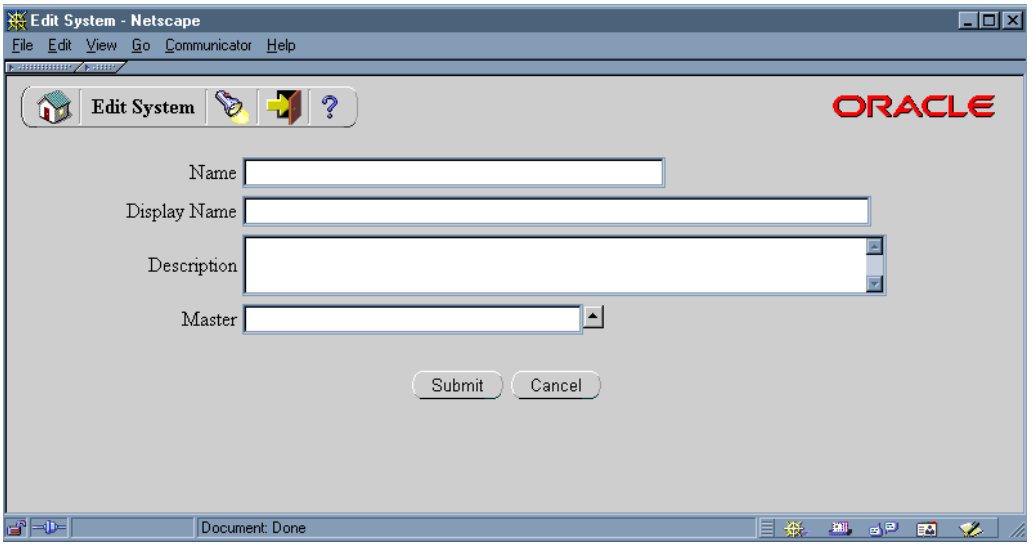
Systems Page



2. The Systems page appears, displaying a list of existing systems. The Systems page summarizes the internal name, display name, and master system of each system. An asterisk marks the local system.

   Choose the Add System button to open the Edit System page.

Edit System Page



3. Enter the internal name of the system in the Name field. All Oracle Workflow APIs, SQL scripts, and PL/SQL procedures refer to the internal name when identifying a system.

   Important: The internal name must be all-uppercase and should not include any single or double quotation marks (' or ") or spaces.

4. Enter a Display Name for the system. This name appears in the Systems list.

5. Enter an optional description for the system.
6. Optionally enter a Master system from which you want this system to receive Event Manager object definition updates. Click on the Master field's up arrow icon to display a list of systems from which to choose. See: Using a List of Values, page 8-17.
7. Choose the Submit button to save the system and return to the Systems page. The Systems page displays an updated list of systems.

   You can also choose the Cancel button to return to the Systems page without saving the system.

To Find Systems:

1. Use a Web browser to connect to the following URL:

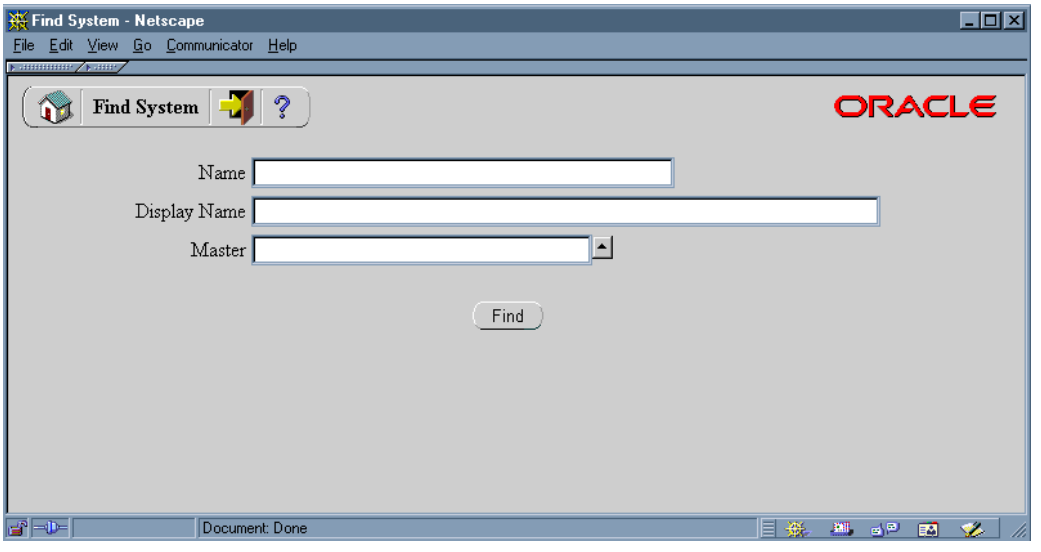
   `<webagent>/wf_event_html.findsystem`

   Replace `<webagent>` with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

   Important: This URL accesses a secured page, so if you have not yet logged on as valid user in the current Web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager Web pages.

   Note: You can also access the Find System Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

   Find System Page



2. The Find System page appears. The Find System page lets you enter search criteria to locate specific systems. The search criteria are:
   - Name - enter the internal name of the system you want to display.

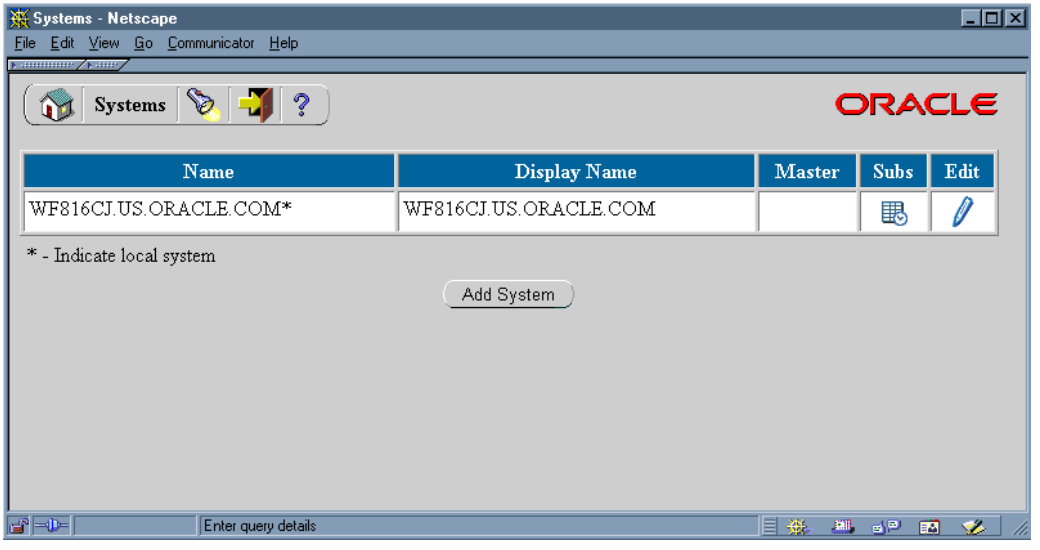
- Display Name - enter the display name of the system you want to display.
- Master - enter the master system for the system you want to display. Click on the field's up arrow icon to display a list of systems from which to choose. See: Using a List of Values, page 8-17.

3. Choose the Find button. The Systems page appears, displaying a list of systems that match your search criteria. An asterisk marks the local system.

To Update or Delete a System:

1. Locate the system you want in the Systems page. You can use the Find System page to find the system that you want and display the system in the Systems page. See: To Find Systems, page 8-15.

Systems Page



2. To view the subscriptions for a system, choose the schedule icon in the Subs column for that system. The Event Subscriptions page appears, displaying the list of subscriptions by the system.

   Note: For systems that do not have any subscriptions yet, a blank schedule icon appears. For systems that do have subscriptions, a full schedule icon appears.

   You can begin defining a new subscription for the system by choosing the Add Subscription button in the Event Subscriptions page. The Edit Subscription page appears with the system name automatically entered in the System field. See Defining Event Subscriptions, page 8-29.

3. To update a system, choose the pencil icon in the Edit column for that system. The Edit System page appears. Make your changes to the system definition and save your work. See: To Define a System, page 8-13.

4. To delete a system, choose the trash icon in the Delete column for that system, and choose OK in the confirmation window that appears. You can also choose Cancel in the confirmation window to return to the Systems page without deleting the system.

Note: You can only delete systems that do not have any agents defined on them or any subscriptions referencing them.

Using a List of Values:

1. For a field that supports a list of values, click on the field's up-arrow icon to display a list of values window.
2. In the Find field, enter search criteria and choose the Find button to retrieve a subset of values that match your criteria. You can also choose the Clear button to clear the Find field. If you do not specify any search criteria and simply choose Find, you retrieve the complete list of values.
3. Click on a value from the list to select that value and close the list of values window. The value you select populates the original field.

Agents

An agent is a named point of communication within a system. Communication within and between systems is accomplished by sending a message from one agent to another. A single system can have several different agents representing different communication alternatives. For example, a system may have different agents to support inbound and outbound communication, communication by different protocols, different propagation frequencies, or other alternatives.

You should define each agent that you will use to communicate events in the Event Manager. Each agent's name must be unique within its system. The agent can be referenced in code within Oracle Workflow by a compound name in the following format:

```
<agent_name>@<system_name>
```

For example, the agent WF_IN within the system HUB could be referenced as `WF_IN@HUB`.

After defining the agents on your local system, you should set them up for event message propagation by scheduling listeners for local inbound agents and propagation for local outbound agents. See: Scheduling Listeners for Local Inbound Agents, *Oracle Workflow Administrator's Guide* and Scheduling Propagation for Local Outbound Agents, *Oracle Workflow Administrator's Guide*.

Assigning a Direction to an Agent

When you define an agent in the Event Manager, you must specify the direction of communication that the agent supports on its local system.

- In - Inbound communication to the system. The agent receives messages in a specific protocol and presents them to the system in a standard format.
- Out - Outbound communication from the system. The agent accepts messages from the system in a standard format and sends them using the specified protocol.

Assigning a Protocol to an Agent

You must associate each agent with the protocol by which it communicates messages. The protocol specifies how the messages are encoded and transmitted. For a message to be successfully communicated, the sending and receiving agents must use the same protocol.

A protocol can represent a network standard, such as SQLNET. It can also represent a business-to-business standard that defines the higher-level message format and handshaking agreements between systems in addition to the network standard.

The Business Event System interacts with an agent through an AQ queue. You can use AQ to perform the propagation of messages by the SQLNET protocol which it supports. In Oracle9*i* Database and higher, AQ also includes Internet access functionality that lets you perform AQ operations over the Internet by using AQ's Internet Data Access Presentation (IDAP) for messages and transmitting the messages over the Internet using transport protocols such as HTTP or HTTPS. Additionally, the Messaging Gateway feature of AQ in Oracle9*i* Database and higher enables communication between applications based on non-Oracle messaging systems and AQ, letting you integrate with third party messaging solutions. You can also implement other services to propagate messages by different protocols.

To implement a custom protocol, you must perform the following steps:

1. Define AQ queues to hold pending inbound and outbound messages.
2. Provide code that propagates messages to and from the AQ queues.
3. Define a lookup code for the new protocol in the Event Protocol Type (`WF_AQ_PROTOCOLS`) lookup type, which is stored in the Standard item type. The Event Manager uses the Event Protocol Type lookup type to validate the protocol for an agent. See: To Create Lookup Codes for a Lookup Type, page 3-21.
4. Define agents with the new protocol. See: To Define an Agent, page 8-25.

If an agent supports inbound communication, you must specify the address by which systems can communicate with it. The format of the address depends on the agent's protocol. For agents that use the SQLNET protocol, the address must be in the following format to enable AQ propagation:

```
<schema>.<queue>@<database link>
```

In this format, *`<schema>`* represents the schema that owns the queue, *`<queue>`* represents the queue name, and *`<database link>`* represents the name of the database link to the instance where the queue is located.

> **Note:** You must enter the database link name exactly as the name was specified when the database link was created. For example, if a database link is named `ORA10.US.ORACLE.COM`, you must enter that complete name in the address of an agent on that database. You cannot abbreviate the name to `ORA10`.
>
> The names of the database links that you want to use for the Business Event System should be fully qualified with the domain names. To confirm the names of your database links, use the following syntax:
>
> ```
> SELECT db_link FROM all_db_links
> ```
>
> See: Creating Database Links, *Oracle Workflow Administrator's Guide*.

Assigning a Queue to an Agent

You must associate each agent on a Workflow-enabled system with an AQ queue. The local system uses this queue to interact with the agent. To send messages, the system enqueues the messages on the queue and sets the recipient addresses. To receive

messages, the system runs a queue listener on the queue. See: Setting Up Queues, *Oracle Workflow Administrator's Guide*.

Event messages within the Oracle Workflow Business Event System are encoded in a standard format defined by the datatype `WF_EVENT_T`. You must assign each agent a PL/SQL package called a queue handler that translates between this standard Workflow format and the format required by the agent's queue. See: Event Message Structure, *Oracle Workflow API Reference*.

> **Note:** Even if the agent's queue uses `WF_EVENT_T` as its payload type, a queue handler is still required in order to set native AQ message properties.

Oracle Workflow provides two standard queue handlers, called WF_EVENT_QH and WF_ERROR_QH, for queues that use SQLNET propagation and use the `WF_EVENT_T` datatype as their payload type. You can use WF_EVENT_QH with queues that handle normal Business Event System processing, while WF_ERROR_QH should be used exclusively with error queues.

Oracle Workflow also provides a standard queue handler called WF_EVENT_OJMSTEXT_QH for queues that use the `SYS.AQ$_JMS_TEXT_MESSAGE` datatype as their payload type. This queue handler enables communication of JMS Text messages through the Business Event System. See: Mapping Between WF_EVENT_T and SYS.AQ$_JMS_TEXT_MESSAGE, *Oracle Workflow API Reference*.

If you want to use queues that require a different format, create a custom queue handler for that format. Your custom queue handler must include a set of standard APIs to enqueue and dequeue messages in the custom format. See: Standard APIs for a Queue Handler, page 6-21.

Standard Agents

When you install Oracle Workflow, several standard agents are automatically defined for the Business Event System.

- WF_CONTROL - Oracle Workflow internal agent, not for customer use
- WF_DEFERRED - Standard agent for deferred subscription processing
- WF_ERROR - Standard agent for error handling
- WF_IN - Default inbound agent
- WF_JMS_IN - Default inbound agent for JMS Text messages
- WF_JMS_OUT - Default outbound agent for JMS Text messages
- WF_NOTIFICATION_IN - Standard inbound agent for e-mail notification responses
- WF_NOTIFICATION_OUT - Standard outbound agent for e-mail notifications
- WF_OUT - Default outbound agent

These agents use standard queues that are automatically defined when you install Oracle Workflow. See: Setting Up Queues, *Oracle Workflow Administrator's Guide*.

You can enable or disable the WF_IN, WF_JMS_IN, WF_OUT, and WF_JMS_OUT agents, but you must not make any other changes to their definitions. You must not make any changes to the definitions of the other agents.

Oracle Workflow automatically runs agent listener service components for the standard WF_DEFERRED, WF_ERROR, and WF_NOTIFICATION_IN agents in order to perform deferred subscription processing, error handling for the Business Event System, and inbound e-mail processing for notification mailers, respectively. If you want to use the WF_IN, WF_JMS_IN, WF_OUT, and WF_JMS_OUT agents for event message propagation, schedule listeners for WF_IN and WF_JMS_IN and propagation for WF_OUT and WF_JMS_OUT as well. You do not need to schedule propagation for the WF_CONTROL and WF_NOTIFICATION_OUT agents, however, because the middle tier processes that use WF_CONTROL dequeue messages directly from its queue, and notification mailers send messages placed on the WF_NOTIFICATION_OUT queue. See: Scheduling Listeners for Local Inbound Agents, *Oracle Workflow Administrator's Guide* and Scheduling Propagation for Local Outbound Agents, *Oracle Workflow Administrator's Guide*.

> **Note:** Oracle Workflow includes three additional agents named WF_RE PLAY_IN, WF_REPLAY_OUT, and WF_SMTP_O_1_QUEUE, which are not currently used. Oracle Workflow also includes agents named WF_JAVA_DEFERRED, WF_JAVA_ERROR, WF_WS_JMS_IN, and WF_WS_JMS_OUT, which are currently used only in Oracle Applications.

The following table lists the default properties for the standard WF_CONTROL agent. See: Cleaning Up the Workflow Control Queue, *Oracle Workflow Administrator's Guide*.

WF_CONTROL Agent Properties

Agent Property	Value
Name	WF_CONTROL
Display Name	Workflow Control Out Queue
Description	Workflow JMS Text Message Queue used to signal messages to middle tier processes
Protocol	SQLNET
Address	*<workflow schema>*.WF_CONTROL@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_CONTROL
Direction	Out
Status	Enabled

The following table lists the default properties for the standard WF_DEFERRED agent.

WF_DEFERRED Agent Properties

Agent Property	Value
Name	WF_DEFERRED
Display Name	WF_DEFERRED
Description	WF_DEFERRED
Protocol	SQLNET
Address	*<workflow schema>*.WF_DEFERRED@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_QH
Queue Name	*<workflow schema>*.WF_DEFERRED
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_ERROR agent.

WF_ERROR Agent Properties

Agent Property	Value
Name	WF_ERROR
Display Name	WF_ERROR
Description	WF_ERROR
Protocol	SQLNET
Address	*<workflow schema>*.WF_ERROR@*<local database>*
System	*<local system>*
Queue Handler	WF_ERROR_QH
Queue Name	*<workflow schema>*.WF_ERROR
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_IN agent.

WF_IN Agent Properties

Agent Property	Value
Name	WF_IN
Display Name	WF_IN
Description	WF_IN
Protocol	SQLNET
Address	*<workflow schema>*.WF_IN@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_QH
Queue Name	*<workflow schema>*.WF_IN
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_JMS_IN agent.

WF_JMS_IN Agent Properties

Agent Property	Value
Name	WF_JMS_IN
Display Name	Workflow JMS In Queue
Description	Workflow JMS Text Message Queue
Protocol	SQLNET
Address	*<workflow schema>*.WF_JMS_IN@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_JMS_IN
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_JMS_OUT agent.

WF_JMS_OUT Agent Properties

Agent Property	Value
Name	WF_JMS_OUT
Display Name	Workflow JMS Out Queue
Description	Workflow JMS Text Message Queue
Protocol	SQLNET
Address	*<workflow schema>*.WF_JMS_OUT@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_JMS_OUT
Direction	Out
Status	Enabled

The following table lists the default properties for the standard WF_NOTIFICATION_IN agent. See: Implementing Notification Mailers, *Oracle Workflow Administrator's Guide*.

WF_NOTIFICATION_IN Agent Properties

Agent Property	Value
Name	WF_NOTIFICATION_IN
Display Name	Workflow Notification In Queue
Description	Workflow inbound notification response queue
Protocol	SQLNET
Address	*<workflow schema>*.WF_NOTIFICATION_IN@ *<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_NOTIFICATION_IN
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_NOTIFICATION_OUT agent. See: Implementing Notification Mailers, *Oracle Workflow Administrator's Guide*.

WF_NOTIFICATION_OUT Agent Properties

Agent Property	Value
Name	WF_NOTIFICATION_OUT
Display Name	Workflow Notification Out Queue
Description	Workflow notification outbound queue
Protocol	SQLNET
Address	*<workflow schema>*.WF_NOTIFICATION_OUT@ *<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_NOTIFICATION_OUT
Direction	Out
Status	Enabled

The following table lists the default properties for the standard WF_OUT agent.

WF_OUT Agent Properties

Agent Property	Value
Name	WF_OUT
Display Name	WF_OUT
Description	WF_OUT
Protocol	SQLNET
Address	*<workflow schema>*.WF_OUT@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_QH
Queue Name	*<workflow schema>*.WF_OUT
Direction	Out
Status	Enabled

Related Topics

To Define an Agent, page 8-25

To Find Agents, page 8-27

To Update or Delete an Agent, page 8-28

Defining Agents

To Define an Agent:

1. Use a Web browser to connect to the following URL:

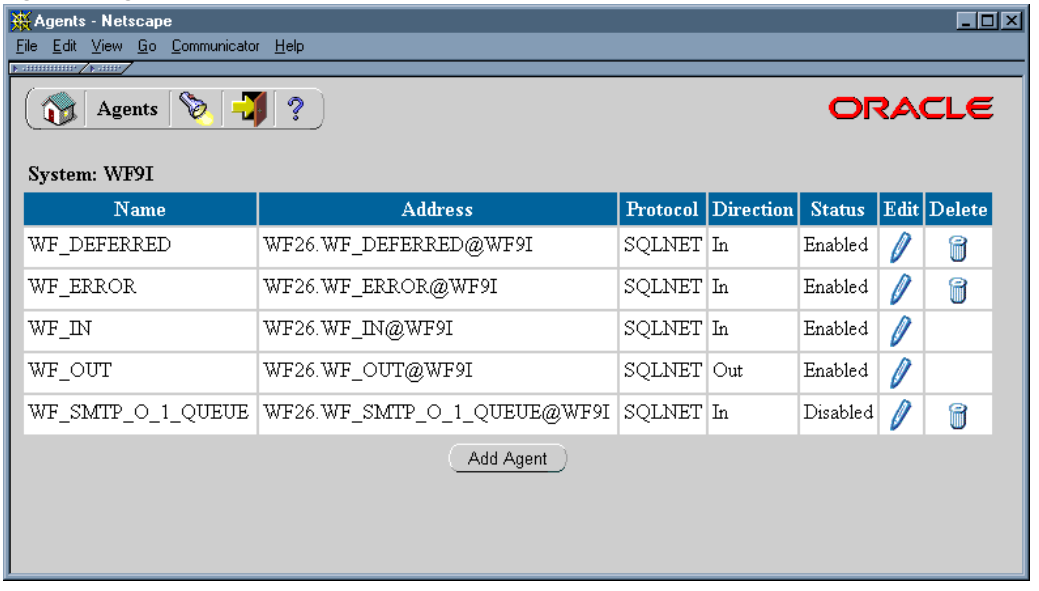
   `<webagent>/wf_event_html.listagents`

   Replace `<webagent>` with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

   > **Important:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current Web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager Web pages.

   > **Note:** You can also access the Agents Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

   Agents Page

Name	Address	Protocol	Direction	Status	Edit	Delete
WF_DEFERRED	WF26.WF_DEFERRED@WF9I	SQLNET	In	Enabled		
WF_ERROR	WF26.WF_ERROR@WF9I	SQLNET	In	Enabled		
WF_IN	WF26.WF_IN@WF9I	SQLNET	In	Enabled		
WF_OUT	WF26.WF_OUT@WF9I	SQLNET	Out	Enabled		
WF_SMTP_O_1_QUEUE	WF26.WF_SMTP_O_1_QUEUE@WF9I	SQLNET	In	Disabled		

2. The Agents page appears, displaying a list of existing agents grouped by the system where they are located. The Agents page summarizes the internal name, address, protocol, direction, and status of each agent.

   Choose the Add Agent button to open the Edit Agent page.

3. Enter the internal name of the agent in the Name field. The agent's internal name must be unique within the agent's system. Oracle Workflow APIs, SQL scripts, and PL/SQL procedures may refer to the internal name when identifying an agent.

   > **Important:** The internal name must be all-uppercase and should not include any single or double quotation marks (`'` or `"`) or spaces.

4. Enter a Display Name for the agent.
5. Enter an optional description for the agent.
6. Select the message communication protocol that the agent supports.
7. If the agent supports inbound communication to its system, enter the address for the agent. The format of the address depends on the protocol you select.

   For agents that use the SQLNET protocol, the address must be in the following format to enable AQ propagation:

   ```
   <schema>.<queue>@<database link>
   ```

   <schema> represents the schema that owns the queue, *<queue>* represents the queue name, and *<database link>* represents the database link to the instance where the queue is located.

   > **Note:** You must enter the database link name exactly as the name was specified when the database link was created. See: Creating Database Links, *Oracle Workflow Administrator's Guide*.

8. Enter the system in which the agent is defined. Click on the System field's up arrow icon to display a list of systems from which to choose. See: Using a List of Values, page 8-17.
9. Enter the Queue Handler for the agent. The queue handler is the PL/SQL package that translates between the Workflow event message format (`WF_EVENT_T`) and the message format required by the queue associated with the agent. See: Standard APIs for a Queue Handler, page 6-21.

   > **Important:** You must enter the queue handler name in all uppercase.

10. Enter the name of the queue that the local system uses to interact with the agent. Since only the local system refers to this queue name, the queue name should be within the scope of this system, without requiring a database link. Use the following format to specify the queue name:

    ```
    <schema>.<queue>
    ```

    <schema> represents the schema that owns the queue, and *<queue>* represents the queue name.

    > **Important:** You must enter the queue name in all uppercase.

11. In the Direction field, select In for an agent that supports inbound communication to its system, or select Out for an agent that supports outbound communication from its system.
12. In the Status field, select Enabled or Disabled as the agent status. If you disable an agent, it still remains in the Agents list for reference, but you cannot use the agent in active subscriptions.
13. Choose the Submit button to save the agent and return to the Agents page. The Agents page displays an updated list of agents.

    You can also choose the Cancel button to return to the Agents page without saving the agent.

To Find Agents:

1. Use a Web browser to connect to the following URL:

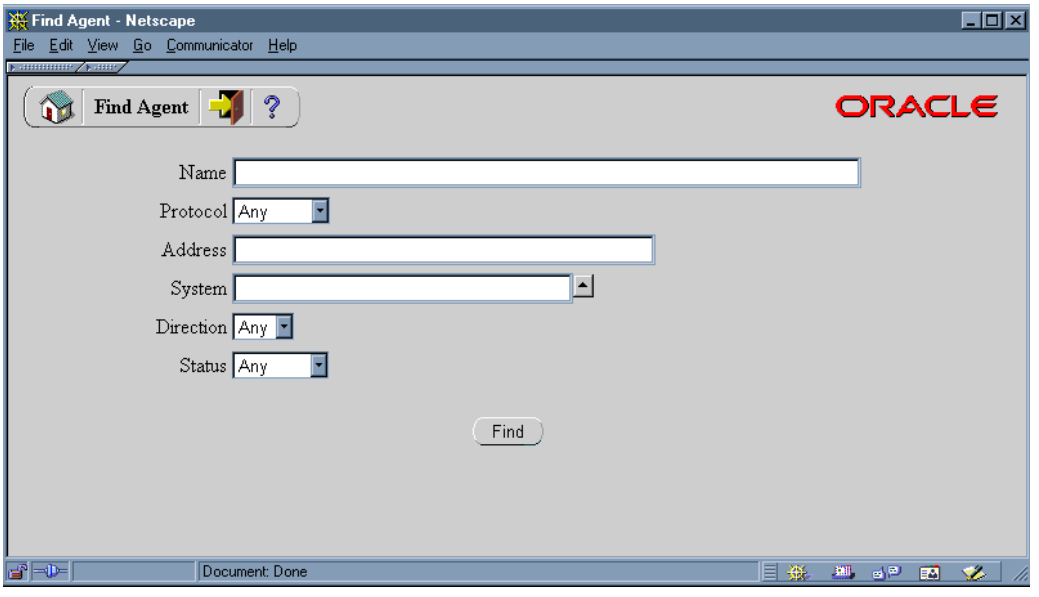
   `<webagent>/wf_event_html.findagent`

   Replace `<webagent>` with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

   > **Important:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current Web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager Web pages.

   > **Note:** You can also access the Find Agent Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

   Find Agent Page

2. The Find Agent page appears. The Find Agent page lets you enter search criteria to locate specific agents. The search criteria are:

   - Name - enter the internal name of the agent you want to display.
   - Protocol - choose the protocol of the agent you want to display, or choose Any to display agents of any protocol.
   - Address - enter the address of the agent you want to display.
   - System - enter the system of the agent you want to display. Click on the field's up arrow icon to display a list of systems from which to choose. See: Using a List of Values, page 8-17.

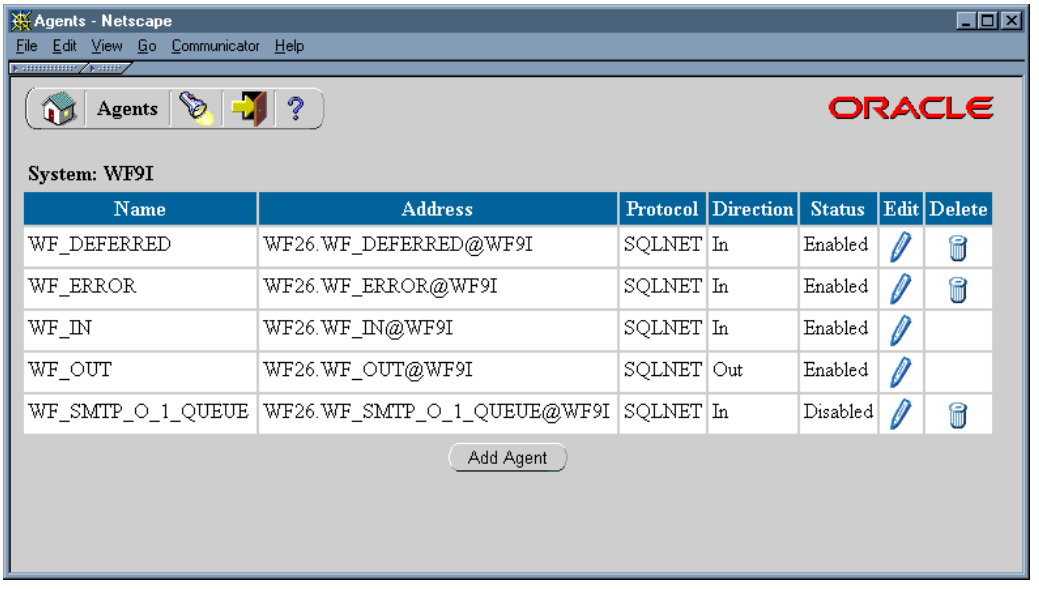
- Direction - choose In or Out as the direction of the agents you want to display, or choose Any to display agents of any direction.
- Status - choose Enabled or Disabled as the status of the agents you want to display, or choose Any to display agents of any status.

3. Choose the Find button. The Agents page appears, displaying a list of agents that match your search criteria.

To Update or Delete an Agent:

1. Locate the agent that you want in the Agents page. You can use the Find Agent page to find the agent that you want and display the agent in the Agents page. See: To Find Agents, page 8-27.

Agents Page



Name	Address	Protocol	Direction	Status	Edit	Delete
WF_DEFERRED	WF26.WF_DEFERRED@WF9I	SQLNET	In	Enabled		
WF_ERROR	WF26.WF_ERROR@WF9I	SQLNET	In	Enabled		
WF_IN	WF26.WF_IN@WF9I	SQLNET	In	Enabled		
WF_OUT	WF26.WF_OUT@WF9I	SQLNET	Out	Enabled		
WF_SMTP_O_1_QUEUE	WF26.WF_SMTP_O_1_QUEUE@WF9I	SQLNET	In	Disabled		

2. To update an agent, choose the pencil icon in the Edit column for that agent. The Edit Agent page appears. Make your changes to the agent definition and save your work. See: To Define an Agent, page 8-25.
3. To delete an agent, choose the trash icon in the Delete column for that agent, and choose OK in the confirmation window that appears. You can also choose Cancel in the confirmation window to return to the Agents page without deleting the agent.

   Note: You can only delete agents that do not have any subscriptions referencing them.

Note: Whenever you make changes to your agents that affect the physical implementation required for message propagation, you should recheck your propagation setup. See: Setting Up the Business Event System, *Oracle Workflow Administrator's Guide*.

Event Subscriptions

An event subscription is a registration indicating that a particular event is significant to a particular system and specifying the processing to perform when the triggering event occurs. You can define your event subscriptions in the Event Manager.

When you install Oracle Workflow, several default subscriptions to predefined Workflow events are automatically created. You can enable, disable, or copy these subscriptions to perform the event processing that you want. See: Predefined Workflow Events, page 9-1.

Whenever an event is raised locally or received from an external source, the Event Manager searches for and executes any eligible subscriptions. To be eligible, a subscription must meet the following requirements:

- The subscriber must be the local system.
- The triggering event must be the event that was raised or received, an event group that includes that event, or the Any event.
- Both the subscription and its triggering event must be marked as active by having a status of Enabled.
- The source type of the subscription must match the source of the event, either local or external.

If no eligible subscriptions exist for the event that occurred (apart from subscriptions to the Any event), then Oracle Workflow executes any enabled subscriptions by the local system to the Unexpected event with the appropriate source type. See: Any Event, page 9-14 and Unexpected Event, page 9-17.

Oracle Workflow provides default error handling for subscription processing through a predefined Error subscription to the Unexpected event and the Default Event Error process in the System: Error item type. You can also define custom error handling for your events. See: Error Handling for Event Subscription Processing, page 11-3.

Defining the Subscriber

To begin defining a subscription, you specify which system is the subscriber. The subscriber is the system where you want the subscription to execute.

Each subscription defines an action on exactly one system, so you should define a separate subscription for each system involved in the processing you want to perform. For example, if you want to propagate data from one system to another, you should define one subscription for the sending system, and another subscription for the receiving system.

Defining How a Subscription is Triggered

You must specify the source type of the events to which the subscription applies. Events can have the following source types:

- Local - The subscription applies only to events raised on the subscribing system.
- External - The subscription applies only to events received by an inbound agent on the subscribing system.

> **Note:** All event messages received by an inbound agent on the subscribing system are considered to have an External

source, whether the sending agent is located on a remote system or on the local system.

- Error - The subscription applies only to errored events dequeued from the WF_ERROR queue.

Next, select the event that you want to trigger the subscription. You can choose either an individual event or an event group. If you choose an event group, the subscription will be triggered whenever any one of the group's member events occurs.

You can also optionally restrict the subscription to be triggered only by events received from a specific source agent. However, in most cases you do not need to specify a source agent.

Controlling How a Subscription is Executed

The phase number for a subscription controls whether the subscription is executed immediately or is deferred. The Event Manager treats subscriptions with a phase number of 100 or higher as deferred subscriptions. Subscriptions with a phase number from 1 to 99 are executed immediately, unless processing for the event is deferred by another method. See: Deferred Subscription Processing, page 8-35.

If you define multiple subscriptions to the same event, the phase numbers for the subscriptions also control the order in which the Event Manager executes those subscriptions. Subscriptions are executed in ascending phase order. For example, you can enter 10 for the subscription that you want to execute first when an event occurs, 20 for the subscription that you want to execute second, and so on. You can use phases to ensure that different types of actions are performed in the appropriate order, such as executing subscriptions that perform validation before subscriptions that perform other types of processing.

> **Note:** If you enter the same phase number for more than one subscription, the Event Manager may execute those subscriptions in any order, relative to each other. However, the Event Manager will still execute that group of subscriptions in their specified place in the phase order, relative to subscriptions with other phase numbers.

Additionally, you can use phase numbers to ensure that your subscriptions are unique, so that the Workflow XML Loader can properly download and upload the subscription definitions. Each subscription must have a unique combination of subscribing system, source type, triggering event, source agent, phase, owner name, and owner tag. If you are defining several subscriptions with similar properties, assign each subscription a different phase number to differentiate it from the others.

The phase number 0 (zero) is reserved for Oracle Workflow seeded subscriptions.

Depending on the processing to be performed, a subscription may require the complete set of event information contained in the event data, or it may require only the event key that identifies the instance of the event. You can improve performance by specifying Key as the rule data for subscriptions that do not require the complete event data. For locally raised events, the Event Manager checks each subscription before executing it to determine whether the subscription requires the complete event data. If the event data is required but is not already provided, the Event Manager runs the generate function for the event to produce the event data. However, if no subscriptions to the event require

the event data, then the Event Manager will not run the generate function, minimizing the resources required to execute the subscriptions.

> **Note:** Even if there are subscriptions that require the complete event data, you can return control to the calling application more quickly after raising the event by deferring all those subscriptions. Then the Event Manager will not run the generate function until those subscriptions are executed at a later time.

Defining the Action for a Subscription

Subscription processing can include the following types of processing:

- Run a function on the event message.
- Send the event message to a workflow process.
- Send the event message to an agent.

Running a Rule Function

To run a function on the event message, you must specify the rule function that you want to execute. You can also specify any additional parameters that you want to pass to the function.

Oracle Workflow provides a standard default rule function named WF_RULE.Default_Rule to perform basic subscription processing. This function is executed by default if no other rule function is specified for the subscription. The default rule function includes the following actions:

- Sending the event message to a workflow process, if specified in the subscription definition
- Sending the event message to an agent, if specified in the subscription definition

See: Default_Rule, *Oracle Workflow API Reference*.

Oracle Workflow also provides additional standard rule functions that you can use for advanced processing, testing, debugging, or other purposes. Commonly used rule functions including the following:

- Default_Rule2 - Performs the basic subscription processing only if the parameter list within the event message includes parameters whose names and values match all the parameters defined for the subscription. See: Default_Rule2, *Oracle Workflow API Reference*.
- Default_Rule3 - Sets the parameter name and value pairs from the subscription parameters into the parameter list within the event message before performing the basic subscription processing. See: Default_Rule3, *Oracle Workflow API Reference*.
- SendNotification - Sends a notification as specified by the parameter list within the event message. This rule function lets you send a notification outside of a workflow process. See: SendNotification, *Oracle Workflow API Reference*.
- Instance_Default_Rule - Sends the event to all existing workflow processes that have eligible receive event activities waiting to receive it, identified by a business key attribute. See: Instance_Default_Rule, *Oracle Workflow API Reference*.

See: Event Subscription Rule APIs, *Oracle Workflow API Reference*.

You can extend your subscription processing by creating custom rule functions. Custom rule functions must be defined according to a standard API. See: Standard API for an Event Subscription Rule Function, page 6-23.

You can use a rule function for many different purposes, including:

- Performing validation
- Processing inbound messages as a receive message handler for an application
- Making modifications to an outbound message, such as adding a correlation ID that associates this message with other messages

> **Note:** You can call *WF_EVENT_FUNCTIONS_PKG.AddCorrelation()* within a custom rule function to add a correlation ID during your custom processing. See: AddCorrelation, *Oracle Workflow API Reference*.

A rule function may read or write to the event message or perform any other database action. However, you should never commit within a rule function. The Event Manager never issues a commit as it is the responsibility of the calling application to commit. Additionally, the function must not change the connection context in any way, including security and NLS settings. If a rule function returns an error, subscription processing is halted.

If the subscription processing that you want to perform for an event includes several successive steps, you may find it advantageous to define multiple subscriptions to the event with simple rule functions that you can reuse, rather than creating complex specialized rule functions that cannot be reused. You can enter phase values for the subscriptions to specify the order in which they should be executed.

> **Note:** If you enter a rule function other than one of the default functions, you can still enter workflow and agent information for your function to reference, but Oracle Workflow does not automatically send the event message to the specified workflow and agent. Instead, you must either explicitly include the send processing in your rule function, or define a separate subscription that does use the default rule function to perform the send processing.

Sending the Event to a Workflow Process

By sending an event to a workflow process, you can model complex processing or routing logic beyond the options of directly running a predefined function or sending the event to a predefined recipient. For example, you can branch to different functions, initiate subprocesses, send notifications, or select recipient agents, based on the contents of the event message, or modify the event message itself.

Events are represented within workflow processes by event activities. See: Event Activity, page 3-62.

To send the event to a particular workflow process using the Default_Rule, Default_Rule2, or Default_Rule3 rule functions, you must specify the item type and process name of the process. The item key for the process is determined either by the correlation ID specified in the event message, or by the event key if no correlation ID is specified. The event can either start a new process with that item key or continue an existing process, identified by that item key, that is waiting to receive it.

Note: The item key for a process instance can only contain single-byte characters. It cannot contain a multibyte value.

Note: You can call *WF_EVENT_FUNCTIONS_PKG.AddCorrelation()* during subscription processing to add a correlation ID to the event message. To use *AddCorrelation(),* you must enter a subscription parameter named ITEMKEY that specifies a function to generate the correlation ID. The function must be specified in the following format:

```
ITEMKEY=<package_name.function_name>
```

See: AddCorrelation, *Oracle Workflow API Reference*.

If you want to send the event to multiple existing workflow processes, instead of to one particular process, you can use the Instance_Default_Rule rule function instead. This rule function does not require you to specify an item type and process name. In this case, the Event Manager sends the event to all existing workflow process instances that have eligible receive event activities waiting to receive it, marked by a business key attribute that matches the event key. For each receive event activity that you want to receive the event, you must define an activity attribute named `#BUSINESS_KEY`, and set an item type attribute as the default value for that activity attribute. Then include logic to set that item attribute value to match the event key of the corresponding event at runtime.

Note: With the Instance_Default_Rule rule function, the event is only sent to continue existing processes. If you want to launch a new process instance with the event, use the Default_Rule, Default_Rule2, or Default_Rule3 rule functions.

If you want to specify additional parameters to set as item attributes for the workflow process, you can enter these parameters in the Parameters field of a subscription and use *WF_RULE.SetParametersIntoParameterList()* in the subscription rule function to set the subscription parameters into the event message parameter list. The event parameters will then be set as item attributes for the workflow process when the process receives the event. See: SetParametersIntoParameterList, *Oracle Workflow API Reference*.

Note: To send an event to a workflow process, you must either use a standard rule function provided by Oracle Workflow or include send processing in your custom rule function. See: Standard API for an Event Subscription Rule Function, page 6-23.

When an event subscription sends an event to a workflow process, the Workflow Engine performs the following processing:

- Sets any parameters in the event message parameter list as item type attributes for the process, creating new item type attributes if a corresponding attribute does not already exist for any parameter.
- Sets the subscription's globally unique identifier (GUID) as a dynamic item attribute so that the workflow process can reference other information in the subscription definition.
- If the event was originally raised by a Raise event activity in another workflow process, the item type and item key for that process are included in the parameter list within the event message. In this case, the Workflow Engine automatically sets

the specified process as the parent for the process that receives the event, overriding any existing parent setting. See: SetItemParent, *Oracle Workflow API Reference*.

- Searches for receive event activities that are eligible to accept the event. For an activity to be eligible, the event must match the activity's event filter, and the activity must either be marked as a Start activity or have a status of 'NOTIFIED', meaning the process has transitioned to the event.

  Note: If you use the Instance_Default_Rule rule function, then to be eligible an activity must have an event filter that matches the event, a status of 'NOTIFIED' , and a business key attribute that matches the event key.

- If an event arrives at a Start activity to launch a new process instance, the Workflow Engine also searches for all other receive event activities that are marked as Start activities and that do not have any incoming transitions, regardless of their event filter. For these activities, the Workflow Engine sets the activity status to 'NOTIFIED' so that they will be ready to receive an event if any more events are sent to this process. This feature lets you design a workflow process that requires multiple events to be received when you do not know in advance the order in which the events will arrive.
- Stores the event name, event key, and event message in item type attributes, as specified in each eligible activity node's event details.
- Marks all the eligible event activity nodes with a status of 'COMPLETED' and continues the thread of execution from each of those nodes.

Sending the Event to an Agent

To send an event to an agent, you must specify either the Out Agent that you want to send the outbound message, or the To Agent that you want to receive the inbound message, or both.

- If you specify both a To Agent and an Out Agent, Oracle Workflow places the event message on the Out Agent's queue for propagation, addressed to the To Agent.
- If you specify a To Agent without an Out Agent, Oracle Workflow selects an outbound agent on the subscribing system whose queue type matches the queue type of the To Agent. The event message is then placed on this outbound agent's queue for propagation, addressed to the To Agent.
- If you specify an Out Agent without a To Agent, Oracle Workflow places the event message on the Out Agent's queue without a specified recipient.
  - You can omit the To Agent if the Out Agent uses a multi-consumer queue with a subscriber list. (The standard Workflow queue handlers work only with multi-consumer queues.) In this case the queue's subscriber list determines which consumers can dequeue the message. If no subscriber list is defined for that queue, however, the event message is placed on the WF_ERROR queue for error handling.

    Note: The subscriber list for a multi-consumer queue in Oracle Advanced Queuing is different from event subscriptions in the Oracle Workflow Business Event System. For more information, see: Subscription and Recipient Lists, *Oracle*

Application Developer's Guide - Advanced Queuing or *Oracle Streams Advanced Queuing User's Guide and Reference*.

- You can also omit the To Agent if the Out Agent uses a single-consumer queue for which you have defined a custom queue handler. For a single-consumer queue, no specified consumer is required.

You can optionally specify the priority with which the recipient should dequeue a message. Messages are dequeued in ascending priority order.

> **Note:** To send an event to an agent, you must either use a standard rule function provided by Oracle Workflow or include send processing in your custom rule function. See: Standard API for an Event Subscription Rule Function, page 6-23.

If you want an event message to become available to the recipient at a future date, rather than being available immediately as soon as it is propagated, you can set the `SEND_DATE` attribute within the event message to the date you want. You should set the send date during subscription processing before the event is sent, either in a prior subscription or earlier in the rule function before the send processing. The event message is propagated to the To Agent but does not become available for dequeuing until the specified date.

Documenting Identifying Information for a Subscription

If you use a program to create subscription definitions automatically, the program can set its own name and brief identifier as the owner name and owner tag for the subscriptions. The program can then use this identifying information to locate the subscriptions that it owns. You can use the Edit Subscription page to update the owner name and owner tag manually if necessary.

Each subscription is assigned a customization level that determines whether you can update the subscription definition. Oracle Workflow uses the customization level to protect Oracle Applications seed data and to preserve your customizations in an upgrade. A subscription can have one of the following customization levels:

- Core - No changes can be made to the subscription definition. This level is used only for subscriptions seeded by Oracle.
- Limit - The subscription status can be updated to Enabled or Disabled, but no other changes can be made to the subscription definition. This level is used only for subscriptions seeded by Oracle.
- User - Any property in the subscription definition can be updated. This level is automatically set for subscriptions that you define.

See: Access Protection for Business Event System Data, *Oracle Workflow Administrator's Guide*.

Deferred Subscription Processing

If you do not want subscriptions for an event to be executed immediately when the event occurs, you can defer the subscriptions. In this way you can return control more quickly to the calling application and let the Event Manager execute any costly subscription processing at a later time.

You can defer subscription processing by three different methods:

- Raise the event with a future date in the `SEND_DATE` attribute. Use this method when you want to defer all subscription processing for a locally raised event until a particular effective date.
- Define subscriptions to the event with phase numbers of 100 or higher. Use this method when you want to defer processing of particular subscriptions for either local or external events.
- Set the dispatch mode of the Event Manager to deferred processing before raising the event. This method can be used to defer all subscription processing for a locally raised event. This method is not recommended, however, and should only be used in exceptional circumstances.

When subscription processing for an event is deferred by any of these methods, the event message is placed on the standard WF_DEFERRED queue associated with the WF_DEFERRED agent. You must ensure a listener is running to monitor the WF_DEFERRED agent. See: Scheduling Listeners for Local Inbound Agents, *Oracle Workflow Administrator's Guide*.

The listener dequeues event messages from the WF_DEFERRED agent in priority order. The event messages retain their original source type, whether Local or External. The amount of time by which subscription processing for these events is deferred depends on the schedule defined for the listener, and, for future-dated events, on the specified effective date.

Deferring Subscription Processing Using a Future Send Date

You can defer subscription processing for a local event until a particular future effective date by raising the event with that date in the `SEND_DATE` attribute. For example, you could enter information for a new employee in a human resources application as soon as the employee was hired, but defer payroll processing until the employee's start date.

When an event is raised with a future send date, the Event Manager immediately places the event message on the WF_DEFERRED queue, without executing any of the subscriptions for the event. All subscriptions to the event are deferred, regardless of their phase number. The event remains in a `WAIT` state until the send date. When the send date arrives, the event message becomes available for dequeuing and will be dequeued the next time an agent listener runs on the WF_DEFERRED queue. The amount of time by which subscription processing is deferred depends on the send date you specify as well as on the schedule defined for the listener.

When the listener dequeues the event message, the Event Manager checks for a subscription ID in the `ERROR_SUBSCRIPTION` attribute. If the event message does not contain a subscription ID, meaning that all subscription processing for the event was deferred immediately after the event was raised, then the Event Manager proceeds to execute all subscriptions to the event, in ascending phase order.

> **Note:** If an event was deferred when it was raised, the Event Manager executes all eligible subscriptions to the event when the event is dequeued from the WF_DEFERRED queue, regardless of the subscription phase numbers. Subscriptions will not be deferred a second time, even if they have a phase number of 100 or higher.

See: Raise, *Oracle Workflow API Reference*.

Deferring Subscription Processing Using Subscription Phase Numbers

You can also use the phase number for a subscription to control whether the subscription is executed immediately or is deferred. The Event Manager treats subscriptions with a phase number of 100 or higher as deferred subscriptions. Both Local and External subscriptions can be deferred in this way.

> **Note:** For this deferral method to take effect when an event is raised locally, the event must not be raised with a future send date, and the Event Manager must be in the normal synchronous mode for subscription processing. Otherwise, the event message will immediately be placed on the WF_DEFERRED queue, and the Event Manager will not execute any subscriptions until the event is dequeued from there.

When a triggering event is raised or received, the Event Manager executes subscriptions to that event in phase order until it encounters a subscription with a phase number of 100 or higher. The Event Manager sets that subscription into the `ERROR_SUBSCRIPTION` attribute within the event message, as well as setting the priority specified in the subscription properties into the `PRIORITY` attribute. Then the event message is placed on the standard WF_DEFERRED queue.

The amount of time by which subscription processing is deferred depends on the schedule defined for the agent listener monitoring the WF_DEFERRED agent. When the listener dequeues an event message, the Event Manager checks for a subscription ID in the `ERROR_SUBSCRIPTION` attribute. If a subscription ID is present, meaning that subscription processing was deferred from that subscription onwards, the Event Manager begins by executing that subscription, and then continues executing any other subscriptions to the event with the same or a higher phase number.

> **Note:** The Event Manager resumes subscription processing at the phase number of the subscription set into the event message. It does not necessarily begin with the phase number 100, if there were no subscriptions with that phase number when the event was originally processed.

Deferring Subscription Processing Using the Event Manager Dispatch Mode

If you raise an event from a local application, you can also choose to defer all subscription processing for that event every single time it is raised. To do so, call the *SetDispatchMode()* API with the mode '`ASYNC`', indicating deferred (asynchronous) processing, just before calling the *Raise()* API. See: SetDispatchMode, *Oracle Workflow API Reference*.

This method is not recommended, however, and should only be used in exceptional circumstances, since it requires hard-coding the deferral in your application. To retain the flexibility to modify subscription processing without intrusion into the application, you can simply raise the event with a future send date or mark some or all of the individual subscriptions for deferral using the subscription phase numbers.

When an event is raised after the dispatch mode is set to deferred processing, the Event Manager immediately places the event message on the WF_DEFERRED queue, without executing any of the subscriptions for the event. All subscriptions to the event are deferred, regardless of their phase number.

The amount of time by which subscription processing is deferred depends on the schedule defined for the agent listener monitoring the WF_DEFERRED agent. When the listener dequeues the event message, the Event Manager checks for a subscription

ID in the `ERROR_SUBSCRIPTION` attribute. If the event message does not contain a subscription ID, meaning that all subscription processing for the event was deferred immediately after the event was raised, then the Event Manager proceeds to execute all subscriptions to the event, in ascending phase order.

> **Note:** If an event was deferred when it was raised, the Event Manager executes all eligible subscriptions to the event when the event is dequeued from the WF_DEFERRED queue, regardless of the subscription phase numbers. Subscriptions will not be deferred a second time, even if they have a phase number of 100 or higher.

Related Topics

To Define an Event Subscription, page 8-38

To Find Event Subscriptions, page 8-42

To Update or Delete an Event Subscription, page 8-43

Standard API for an Event Subscription Rule Function, page 6-23

Scheduling Listeners for Local Inbound Agents, *Oracle Workflow Administrator's Guide*

Defining Event Subscriptions

To Define an Event Subscription:

1. Use a Web browser to connect to the following URL:

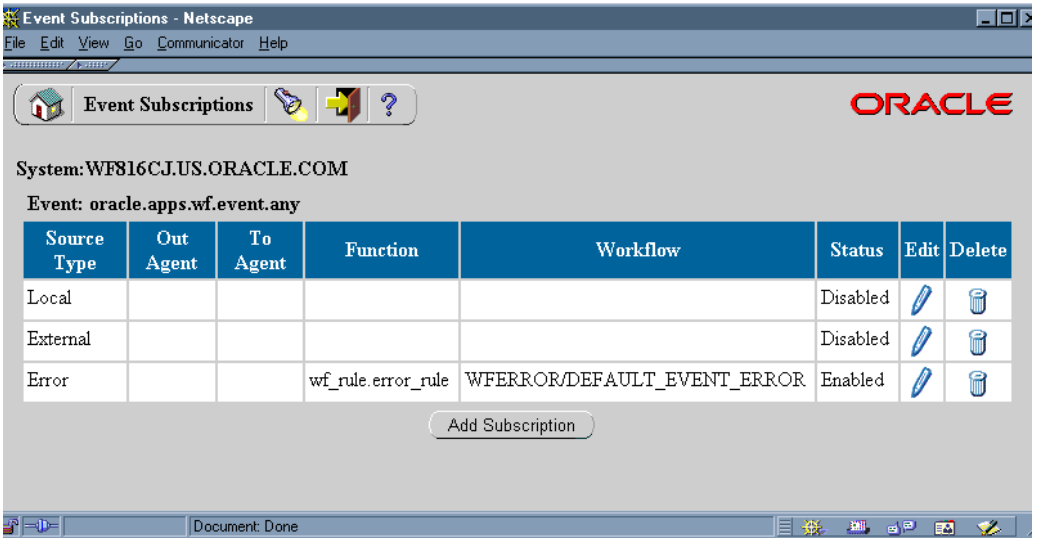
   ```
   <webagent>/wf_event_html.listsubscriptions
   ```

   Replace `<webagent>` with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

   > **Important:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current Web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager Web pages.

   > **Note:** You can also access the Event Subscriptions Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

Event Subscriptions Page



2. The Event Subscriptions page appears, displaying a list of existing subscriptions grouped by the subscribing system and triggering event. The Event Subscriptions page summarizes the source type, out agent, to agent, function, workflow, and status of each subscription.

   Choose the Add Subscription button to open the Edit Subscription page.

   > **Note:** For seeded subscriptions with a customization level of Limit, you can only update the subscription status. For seeded subscriptions with a customization level of Core, you cannot update any properties; you can only view the subscription definition.

3. In the Subscriber region, enter the system where the subscription executes. Click on the System field's up arrow icon to display a list of systems from which to choose. See: Using a List of Values, page 8-17.

4. In the Triggering Condition region, specify the type of source system to which the subscription applies in the Source Type field.

   - Local - The subscription applies only to events raised on the subscribing system.
   - External - The subscription applies only to events received by an inbound agent on the subscribing system.

     > **Note:** All event messages received by an inbound agent on the subscribing system are considered to have an External source, whether the sending agent is located on a remote system or on the local system.

   - Error - The subscription applies to only to errored events dequeued from the WF_ERROR queue.

5. Enter the event to which the subscription applies in the Event Filter field. You can specify an individual event or an event group. Click on the Event Filter field's up arrow icon to display a list of events from which to choose. See: Using a List of Values, page 8-17.

6. Enter an optional Source Agent to which the subscription applies. If you specify a source agent, then the subscription is executed only when the triggering event is received from that agent. Click on the Source Agent field's up arrow icon to display a list of agents from which to choose. See: Using a List of Values, page 8-17.

   Note: In most cases, the Source Agent field is left blank.

7. In the Execution Control region, enter a phase number for the subscription to specify the order in which subscriptions that apply to the same event are executed. The phase number also controls whether a subscription is executed immediately or is deferred, unless processing for the event is deferred by another method.

   Note: Each subscription must have a unique combination of subscribing system, source type, triggering event, source agent, phase, owner name, and owner tag. If you are defining several subscriptions with similar properties, assign each subscription a different phase number to differentiate it from the others.

8. Select Enabled or Disabled as the subscription status. If you disable a subscription, it still remains in the Event Subscriptions list for reference, but it can no longer be actively used to respond to events.
9. In the Rule Data field, specify the event information required by the subscription.
   - Key - The subscription requires only the event key.
   - Message - The subscription requires the complete event data.
10. In the Action region, define the subscription processing you want to perform when the triggering event occurs. Subscription processing can include:
    - Running a function on the event message.
    - Sending the event message to a workflow process.
    - Sending the event message to an agent.
11. If you want to run a function on the event message, enter the Rule Function to run. The rule function must be defined according to a standard API. See: Standard API for an Event Subscription Rule Function, page 6-23.

    If you do not specify a rule function, Oracle Workflow runs a default rule function to send the event message to the workflow process and the agent that you specify.

    Note: If you enter a rule function other than the default, Oracle Workflow does not automatically send the event message to the specified workflow and agent. You must explicitly include the send processing in your custom rule function instead. You can still enter workflow and agent information in the Action region for your function to reference, however.

12. If you want to send the event message to a workflow process, enter the item type that the process belongs to in the Workflow Item Type field and the name of the process in the Workflow Process Name field. Click on each field's up arrow icon to display a list of values from which to choose. See: Using a List of Values, page 8-17.

Note: The list of values for the Workflow Process Name field includes only the runnable processes within the item type you specify.

13. If you want to send the event message to an agent, enter either the Out Agent that you want to send the outbound message, or the To Agent that you want to receive the inbound message, or both. Click on each field's up arrow icon to display a list of values from which to choose. See: Using a List of Values, page 8-17.
    - If you specify both a To Agent and an Out Agent, Oracle Workflow places the event message on the Out Agent's queue for propagation, addressed to the To Agent.
    - If you specify a To Agent without an Out Agent, Oracle Workflow selects an outbound agent on the subscribing system whose queue type matches the queue type of the To Agent. The event message is then placed on this outbound agent's queue for propagation, addressed to the To Agent.
    - If you specify an Out Agent without a To Agent, Oracle Workflow places the event message on the Out Agent's queue without a specified recipient. The Out Agent must use either a multi-consumer queue with a subscriber list or a single-consumer queue.

      Note: The Out Agent must be located on the subscribing system. The list of values for the Out Agent field includes only agents with a direction of Out.

      The list of values for the To Agent field includes only agents with a direction of In.

14. If you want to send the event message to an agent, select Normal, High, or Low as the priority with which the recipient should dequeue the message.
15. Optionally enter any additional parameters for the rule function in the Parameters field. Use spaces to separate the parameters, and specify the name and value for each parameter in the following format:

    ```
    <name1>=<value1> <name2>=<value2> ... <nameN>=<valueN>
    ```

    Note: If you send the event message to a workflow process and you want to specify additional parameters to set as item attributes for the process, you can enter these parameters in the Parameters field for a subscription and use *WF_RULE.SetParametersIntoParameterList()* in the subscription rule function to set the subscription parameters into the event message parameter list. The event parameters will then be set as item attributes for the workflow process when the process receives the event. See: SetParametersIntoParameterList, *Oracle Workflow API Reference*.

16. In the Documentation region, you can optionally identify the program or application that owns the subscription by entering the program name in the Owner Name field and the program ID in the Owner Tag field. The Owner Name and Owner Tag are not required if you are defining a subscription manually in the Edit Subscription page. However, if you use a program to create subscription definitions automatically, the Event Manager displays the owner information set by that

program in these fields. You can use the Edit Subscription page to update this information manually if necessary.

17. You can review the customization level for the subscription.
    - Core - No changes can be made to the subscription definition. This level is used only for subscriptions seeded by Oracle.
    - Limit - The subscription status can be updated to Enabled or Disabled, but no other changes can be made to the subscription definition. This level is used only for subscriptions seeded by Oracle.
    - User - Any property in the subscription definition can be updated. This level is automatically set for subscriptions that you define.
18. Enter an optional description for the subscription.
19. Choose the Submit button to save the subscription and return to the Event Subscriptions page. The Event Subscriptions page displays an updated list of subscriptions.

    You can also choose the Cancel button to return to the Event Subscriptions page without saving the subscription.

To Find Event Subscriptions:

1. Use a Web browser to connect to the following URL:

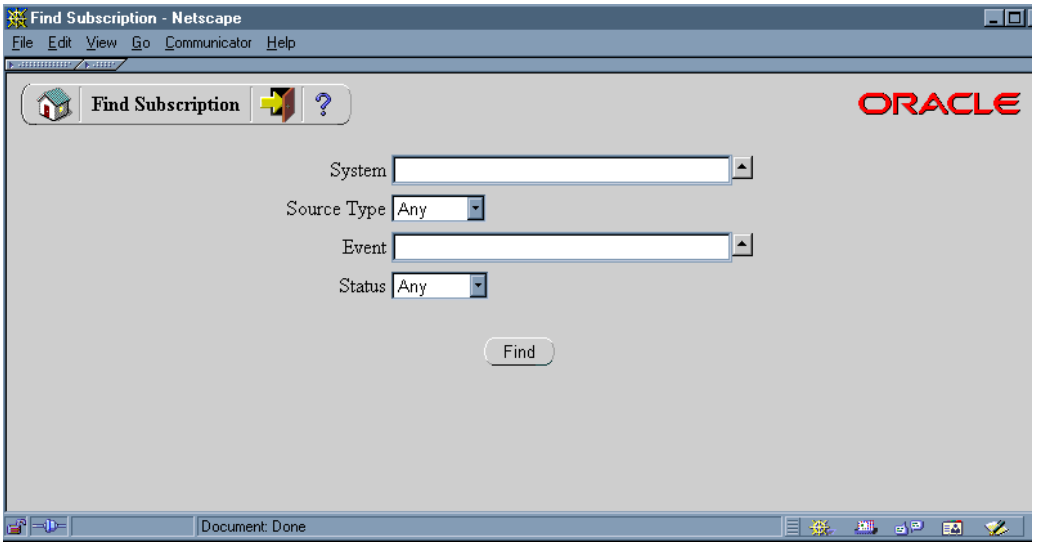
   `<webagent>/wf_event_html.findsubscription`

   Replace `<webagent>` with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

   > **Important:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current Web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager Web pages.

   > **Note:** You can also access the Find Subscription Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

Find Subscription Page



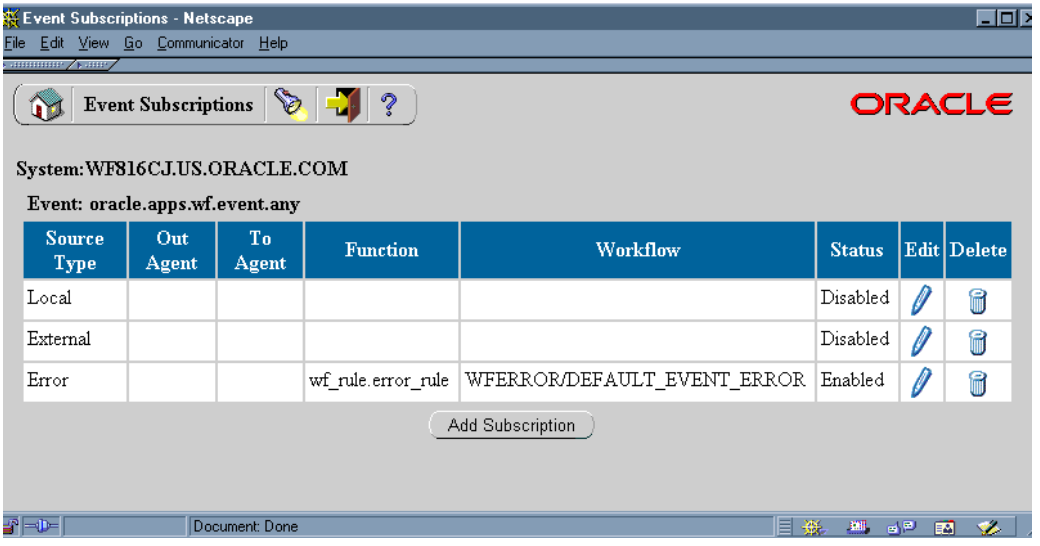
2. The Find Subscription page appears. The Find Subscription page lets you enter search criteria to locate specific event subscriptions. The search criteria are:
   - System - select the system for which you want to display subscriptions. Click on the field's up arrow icon to display a list of systems from which to choose. See: Using a List of Values, page 8-17.
   - Source Type - choose Local, External, or Error as the type of source system for which you want to display subscriptions, or choose Any to display subscriptions for any type of source system.
   - Event - select the event for which you want to display subscriptions. Click on the field's up arrow icon to display a list of events from which to choose. See: Using a List of Values, page 8-17.
   - Status - choose Enabled or Disabled as the status of the subscriptions you want to display, or choose Any to display subscriptions of any status.
3. Choose the Find button. The Event Subscriptions page appears, displaying a list of subscriptions that match your search criteria.

   If your search criteria included a system or an event, you can choose the Add Subscription button to open the Edit Subscription page with the system and event information you specified automatically entered in the System and Event Filter fields, respectively. You can begin defining a new subscription in this way even if your search did not find any existing subscriptions matching your criteria.

To Update or Delete an Event Subscription:

1. Locate the subscription you want in the Event Subscriptions page. You can use the Find Subscription page to find the subscription that you want and display the subscription in the Event Subscriptions page. See: To Find Event Subscriptions, page 8-42.

Event Subscriptions Page



2. To update a subscription, choose the pencil icon in the Edit column for that subscription. The Edit Subscription page appears. Make your changes to the subscription definition and save your work. See: To Define an Event Subscription, page 8-38.

   Note: For seeded subscriptions with a customization level of Limit, you can only update the subscription status. For seeded subscriptions with a customization level of Core, you cannot update any properties; you can only view the subscription definition.

3. To delete a subscription, choose the trash icon in the Delete column for that subscription, and choose OK in the confirmation window that appears. You can also choose Cancel in the confirmation window to return to the Event Subscriptions page without deleting the subscription.

   Note: You cannot delete any subscription seeded by Oracle that has a customization level of Core or Limit.

Raising Events

In addition to raising events from your applications or through workflows, you can raise events that do not require additional parameters manually using the Raise Event Web page for testing purposes. When you raise an event, the Event Manager searches for and executes any enabled subscriptions by the local system to that event with a source type of Local, and also any enabled subscriptions by the local system to the Any event with a source type of Local.

To Raise an Event:

1. Use a Web browser to connect to the following URL:

   `<webagent>/wf_event_html.entereventdetails`

Replace `<webagent>` with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

> **Important:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current Web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager Web pages.

> **Note:** You can also access the Raise Event Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

2. The Raise Event page appears.
3. In the Event Name field, select the event that you want to raise.
4. Enter an event key that uniquely identifies this instance of the event.
5. Optionally enter event data to describe what occurred in the event.

   > **Note:** The maximum length of the data you can enter in the Event Data field is 32 kilobytes. If the event data exceeds 32 Kb, you should assign a generate function in the event definition to generate the event data, rather than entering the data directly in the Event Data field. See: To Define an Event, page 8-5.
   >
   > You can also choose to raise the event using the WF_EVENT.Raise API instead. This method lets you provide the event data as a CLOB storing up to four gigabytes of data. See: Raise, *Oracle Workflow API Reference*.

6. Choose the Submit button to raise the event to the Event Manager. You can also choose the Cancel button to return to the Oracle Workflow home page without raising the event.

   If you choose the Submit button, Oracle Workflow raises the event and displays a confirmation message with the event name and event key. Choose the OK button to return to the Raise Event page.

Signing Up Systems

Before you can send business events from one system to another, you must sign up the destination system with the source system as a potential recipient of event messages. Signing up a system means defining the destination system as well as its inbound agents in the Event Manager of the source system, so that event messages from the source system can be addressed to the destination agents.

Usually, both systems should be signed up with each other, so that each system can both send messages to and receive messages from the other system.

To sign up a destination system for receiving event messages from a source system, perform the following steps:

1. Retrieve the local system and inbound agent definitions, which together make up the system identifier information, from the destination system. You can use the System Identifier Web page on the destination system to generate an XML document containing the system identifier information. See: To Retrieve System Identifier Information, page 8-46.

   Note: If you do not have access to the Oracle Workflow installation on the destination system, ask the workflow administrator for that system to perform this step.

2. Add the destination system identifier information to the Event Manager in the source system. You can use the System Signup Web page on the source system to add the information by raising the System Signup event with the XML document from the destination system as the event data. When the System Signup event is raised on the source system, Oracle Workflow executes a predefined subscription that adds the system identifier information to the Event Manger in that system. See: To Sign Up a System, page 8-47.

   Note: If you do not have access to the Oracle Workflow installation on the source system, ask the workflow administrator for that system to perform this step.

To Retrieve System Identifier Information:

1. Use a Web browser to connect to the following URL on the system you want to sign up as a destination system:

   `<webagent>/wf_event_html.getsystemidentifier`

   Replace `<webagent>` with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

   Important: This URL accesses a secured page, so if you have not yet logged on as a valid user in the current Web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager Web pages.

   If you do not have access to the Oracle Workflow installation on the destination system, ask the workflow administrator for that system to perform this step.

   Note: You can also access the System Identifier Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

2. Oracle Workflow produces the system identifier XML document, which contains the definitions of the local system and its inbound agents. Save this document as a text file. You can then copy the document and enter it as the event data for the System Signup event when you sign this system up with a source system. See: To Sign Up a System, page 8-47.

To Sign Up a System:

1. Use a Web browser to connect to the following URL on the source system where you want to sign up a destination system:

   ```
   <webagent>/wf_event_html.entereventdetails?p_event_name=
   oracle.apps.wf.event.system.signup
   ```

   Replace *<webagent>* with the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

   > **Important:** This URL accesses a secured page, so if you have not yet logged on as valid user in the current Web session, you will be prompted to do so before the page appears. You must have workflow administrator privileges to access the Event Manager Web pages.
   >
   > If you do not have access to the Oracle Workflow installation on the source system, ask the workflow administrator for that system to perform this step.
   >
   > **Note:** You can also access the System Signup Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

2. The System Signup page appears, displaying the internal name of the System Signup event.
3. Enter an event key that uniquely identifies this instance of the event.
4. Copy the XML document containing the destination system identifier information into the Event Data field. See: To Retrieve System Identifier Information, page 8-46.
5. Choose the Submit button to raise the System Signup event to the Event Manager. A confirmation message is displayed. When the System Signup event is raised, Oracle Workflow executes a predefined subscription that adds the system identifier information from the event data to the Event Manager. See: System Signup Event, page 9-13.

   You can also choose the Cancel button to return to the Oracle Workflow home page without raising the System Signup event.

Synchronizing Systems

Synchronizing systems means replicating all the Event Manager objects that are defined on the source system to the target system. You can use the Synchronize Event Systems event to synchronize systems with each other.

To Synchronize Systems:

1. Sign up the source and target systems with each other. See: Signing Up Systems, page 8-45.
2. On the source system, copy the predefined subscription to the Seed Event Group with the Local source type and modify the copy as follows.

- Specify the inbound agent on the target system that you want to receive the event message, or specify a workflow process that sends the event message to the target system.

  Note: If you want to send the event message to more than one target system, you can either define additional subscriptions or specify a workflow process that sends the event message to multiple systems.

- Enable the subscription.

  Note: If you do not have access to the Oracle Workflow installation on the source system, ask the workflow administrator for that system to perform this step.

3. On the target system, enable the predefined subscription to the Seed Event Group with the External source type.

   Note: If you do not have access to the Oracle Workflow installation on the target system, ask the workflow administrator for that system to perform this step.

4. On the source system, raise the Synchronize Event Systems event (`oracle.apps.wf.event.all.sync`) using the Raise Event Web page. Enter a unique event key, but leave the Event Data field blank. See: Raising Events, page 8-44

   Note: If you do not have access to the Oracle Workflow installation on the source system, ask the workflow administrator for that system to perform this step.

When the Synchronize Event Systems event is raised on the source system, it triggers the subscription to the Seed Event Group with the Local source type. The Event Manager generates the event message, which contains the definitions of all the Event Manager objects on the local system, including events, event groups, systems, agents, and subscriptions. Then the event message is sent to the specified inbound agent on the target system, or to the specified workflow process that sends the event message to the target system.

When the Synchronize Event Systems event is received on the target system, it triggers the subscription to the Seed Event Group with the External source type. Oracle Workflow loads the object definitions from the event message into the Event Manager on the target system, creating new definitions or updating existing definitions as necessary.

Automatic Replication:

After you enable the subscriptions in steps 2 and 3, these subscriptions will also replicate any subsequent changes you make to Event Manager object definitions on the source system. Whenever you create, update, or delete events, event group members, systems, agents, or subscriptions, Oracle Workflow raises the corresponding predefined events. These events trigger the Local subscription to the Seed Event Group on the source system, which sends the object definition data to the target system. The External subscription to the Seed Event Group on the target system receives the data and adds or updates the object definition in the Event Manager there.

If you do not want to continue automatically replicating changes on the source system to the target system, you can either disable the subscriptions after you finish synchronizing the systems, or disable the predefined events corresponding to those changes.

Master/Copy Systems:

If you choose, you can treat one system as a master system that replicates its own Event Manager object definitions to its associated copy systems, but does not accept any object definition changes from those systems. To set up master/copy replication, perform the steps to synchronize the target copy systems with the source master system, as usual. Then, to prevent object definitions from being sent from the copy systems, ensure that the Local subscription to the Seed Event Group on the copy systems is disabled. To prevent object definitions from being received into the master system, ensure that the External subscription to the Seed Event Group on the master system is disabled as well.

Related Topics

Predefined Workflow Events, page 9-1

Synchronize Event Systems Event, page 9-9

Seed Event Group, page 9-9

To Define an Event Subscription, page 8-38

Event Manager for Oracle Applications

The Oracle Workflow Event Manager lets you register interesting business events that may occur in your applications, the systems among which events will be communicated, named communication agents within those systems, and subscriptions indicating that an event is significant to a particular system. The Event Manager also performs subscribtion processing when events occur.

You must have workflow administrator privileges to maintain Business Event System objects in the Event Manager pages. If you have workflow administrator privileges, you can create, update, and delete events, subscriptions, systems, and agents. You can also test business events by manually raising a test event. If you do not have administrator privileges, you can view Business Event System objects in the Event Manager, but you cannot modify them. Workflow administrator privileges are assigned in the Workflow Configuration page. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

> **Note:** When defining internal names for Business Event System objects in the Event Manager, use only characters from the ASCII character set.

You can use the Workflow XML Loader to upload and download XML definitions for Business Event System objects between a database and a flat file. See: Using the Workflow XML Loader, *Oracle Workflow Administrator's Guide*.

When an event occurs in an application on your local system, an event key must be assigned to uniquely identify that particular instance of the event. Then the event must be raised to the Event Manager.

You can raise an event by any of the following methods:

- Raise the event from the application where the event occurs using the *WF_EVENT.Raise()* PL/SQL API or the raise method in the BusinessEvent Java class. See: Raise, *Oracle Workflow API Reference* and the Oracle Workflow Javadoc.
- Raise the event from a workflow process using a Raise event activity. See: Event Activity, page 3-62.
- Raise the event manually using the Test Business Event page. See: To Raise a Test Event, page 8-57.

Additionally, the Event Manager can receive events sent from the local system or remote systems.

When an event is raised by a local application or received from a local or external system, the Event Manager executes any subscriptions to that event. Depending on the action defined in the subscription, the Event Manager may send the event information to a workflow process, send the event information to an agent, send a notification, send or receive an Oracle XML Gateway message, or execute custom code.

To communicate event messages between systems, you must schedule propagation for outbound messages and run agent listeners for inbound messages. You can use Oracle Enterprise Manager to schedule propagation and the Workflow Manager component of Oracle Applications Manager to run agent listeners. You can also view the distribution of event messages on different agents in Workflow Manager, drill down to view details about individual event messages, and review queue details for the agents. For more information, please refer to the Oracle Applications Manager online help, Oracle Enterprise Manager online help, Oracle Enterprise Manager Support, *Oracle Application Developer's Guide - Advanced Queuing* or *Oracle Streams Advanced Queuing User's Guide and Reference*, Distributed Management, *Oracle Enterprise Manager Administrator's Guide*, and Setting Up the Business Event System, *Oracle Workflow Administrator's Guide*.

To test your Business Event System setup, you can run the Workflow Agent Ping/Acknowledge workflow. See: Workflow Agent Ping/Acknowledge, page 8-99.

> **Note:** For information about the version of the Event Manager available in standalone Oracle Workflow, see: Event Manager for Standalone Oracle Workflow, page 8-2.

Related Topics

Events, page 8-50

Event Subscriptions, page 8-58

Agents, page 8-76

Systems, page 8-92

Events

A business event is an occurrence in an internet or intranet application or program that might be significant to other objects in a system or to external agents. For instance, the creation of a purchase order is an example of a business event in a purchasing application. You can define your significant events in the Event Manager.

Oracle Workflow provides several predefined events for significant occurrences within the Business Event System. See: Predefined Workflow Events, page 9-1.

When you define an event in the Event Manager, you must assign it a unique internal name, which is case-sensitive. The suggested format for these internal names is a compound structure of identifiers separated by periods (`.`) as follows:

```
<company>.<family>.<product>.<component>.<object>.<event>
```

This format allows you to organize the events you define into a classification hierarchy.

Any detail information needed to describe what occurred in an event, in addition to the event name and event key, is called the event data. For example, the event data for a purchase order event includes the item numbers, descriptions, and cost. The event data can be structured as an XML document.

The application where the event occurs can include the event data when raising the event to the Event Manager. If the application will not provide the event data, you should specify a generate function for the event that can produce the complete event data from the event name, event key, and an optional parameter list. The generate function must follow a standard PL/SQL or Java API. See: Raise, *Oracle Workflow API Reference* and Standard API for an Event Data Generate Function, page 6-19.

Define only one generate function for an event, either PL/SQL or Java. Define the generate function independently of any subscriptions to the event.

- PL/SQL generate function - The Business Event System performs PL/SQL subscription processing in the database tier for an event with a PL/SQL generate function. However, if subscription processing for the event moves to the middle tier due to Java subscriptions, the middle tier Business Event System will execute the PL/SQL generate function through Java Database Connectivity (JDBC), if required.
- Java generate function - The Business Event System will always perform subscription processing for the event in the middle tier to enable execution of the Java generate function, even if the subscriptions to the event are all PL/SQL subscriptions. If the event is raised from PL/SQL code or received through a PL/SQL agent listener, the Event Manager places the event on the WF_JAVA_DEFERRED queue to move subscription processing to the middle tier.

  > **Note:** When raising an event from Java, you can also set a serializable object as a payload for the event. The payload is available to all Java subscriptions to the event.

The Event Manager checks each subscription before executing it to determine whether the subscription requires the event data. If the event data is required but is not already provided, the Event Manager calls the generate function for the event to produce the event data. If the event data is required but no generate function is defined for the event, Oracle Workflow creates a default set of event data using the event name and event key.

> **Note:** If the generate function is costly, and you want to return control to the calling application more quickly after raising the event, you can defer all the subscriptions that require the complete event data. Then the Event Manager will not run the generate function until those subscriptions are executed at a later time. See: Deferred Subscription Processing, page 8-66.

You can run a diagnostic test through Oracle Diagnostics to verify that the generate functions that are defined for events exist in the database and are valid. See: Oracle Workflow Diagnostic Tests, *Oracle Workflow Administrator's Guide*.

You can associate an event with the program or application to which it belongs by setting the program name and brief identifier as the owner name and owner tag for the event. For instance, in the example of the purchase order event, the owner would be the purchasing application. The program can then use this identifying information to locate the events that it owns.

Each event is assigned a customization level that determines whether you can update the event definition. Oracle Workflow uses the customization level to protect Oracle Applications seed data and to preserve your customizations in an upgrade. An event can have one of the following customization levels:

- Core - No changes can be made to the event definition. This level is used only for events seeded by Oracle Applications.
- Limit - The event status can be updated to Enabled or Disabled, but no other changes can be made to the event definition. This level is used only for events seeded by Oracle Applications.
- User - Any property in the event definition can be updated. This level is automatically set for events that you define.

See: Access Protection for Business Event System Data, *Oracle Workflow Administrator's Guide*.

Some Oracle Applications products provide seeded events and subscriptions. In these cases, Oracle Workflow executes subscriptions only if the triggering event and the subscription are both owned by products that you have licensed with a status of Installed or Shared.

Your Oracle Applications installation may include seeded events owned by Oracle Applications products that you have not licensed. For such events, the Update Event page displays a notice that the event is not licensed. Oracle Workflow will not execute any subscriptions to these events. Additionally, Oracle Workflow will not execute any subscriptions owned by products that you have not licensed, even if the triggering events for those subscriptions are licensed.

You can use the License Manager AD utility to review which products you currently have licensed. To ensure that the license status of the seeded events and subscriptions in the Business Event System is updated according to the status of the products you currently have licensed, you can run the Synchronize Product License and Workflow BES License concurrent program. See: License Manager, *Oracle Applications AD Utilies Reference Guide* and Synchronizing License Statuses, *Oracle Workflow Administrator's Guide*.

> **Note:** Any events that you define with a customization level of User are always treated as being licensed.

Event Groups

You can also define event groups that let you associate any events you want with each other and reference them as a group in event subscriptions. An event group is a type of event composed of a set of individual member events. The internal names of event groups should follow the same format as the names of individual events. Once you have defined an event group, you can register a subscription to the group rather than having

to create separate subscriptions for each individual event within it. The subscription will be executed whenever any one of the group's member events occurs.

> **Note:** Event groups cannot be used to raise events. You must raise each event individually.

Related Topics

To View and Maintain Events, page 8-53

To Create or Update an Event, page 8-54

To Create or Update an Event Group, page 8-55

To Raise a Test Event, page 8-57

Defining Events

To View and Maintain Events:

1. Use a Web browser to navigate to the Event Manager, using a responsibility and navigation path specified by your system administrator. Then choose Events in the horizontal navigation. See: Oracle Workflow Developer Navigation Paths, page A-1.
2. Search for the events you want to display. The main search option is:
   - Name - Enter the internal name of the event you want to display. You can enter a partial value to search for events whose internal names contain that value.

   You can also enter the following additional search criteria:
   - Display Name - Enter the display name of the event you want to display.
   - Type - Select Event or Group as the type of the events you want to display.
   - Status - Select Enabled or Disabled as the status of the events you want to display.
   - Owner Name - Enter the name of the program or application that owns the event you want to display.
   - Owner Tag - Enter the program ID of the program or application that owns the event you want to display.
3. To view the subscriptions to an event, select the subscription icon for that event.

   > **Note:** For events that do not have any subscriptions, a blank subscription icon appears in the Subscription column. For events that do have subscriptions to them, a full subscription icon appears.

   - If you have workflow administrator privileges, you can define a new subscription to the event by selecting the Create Subscription button. The Create Event Subscription page appears with the event name automatically entered in the Event Filter field. See: To Create or Update a Subscription, page 8-71.
   - If you have workflow administrator privileges, you can update an existing subscription by selecting the update icon for that subscription. See: To Create or Update a Subscription, page 8-71.

     > **Note:** For seeded subscriptions with a customization level of Limit, you can only update the subscription status. For seeded

subscriptions with a customization level of Core, you cannot update any properties; you can only view the subscription definition.

4. If you have workflow administrator privileges, you can use the administration icons and buttons in the Events page to perform administrative operations.
   - To update an event, choose the update icon for that event. See: To Create or Update an Event, page 8-54 and To Create or Update an Event Group, page 8-55.

     Note: For seeded events with a customization level of Limit, you can only update the event status. For seeded events with a customization level of Core, you cannot update any properties; you can only view the event definition.

   - To manually raise a test event, select the test icon for the event you want. See: To Raise a Test Event, page 8-57.

     Note: You cannot raise an event group. You must raise each event individually.

   - To delete an event, select the event and select the Delete button.

     Note: You can only delete events that do not have any subscriptions referencing them and that do not belong to any event groups.

     Additionally, you cannot delete any event seeded by Oracle Applications that has a customization level of Core or Limit.

   - To create a new individual event, select the Create Event button. See: To Create or Update an Event, page 8-54.
   - To create a new event group, select the Create Event Group button. See: To Create or Update an Event Group, page 8-55.

To Create or Update an Event:

1. Navigate to the Create Event page or to the Update Event page. The Create Event page and the Update Event page are identical, except that the fields in the Update Event page are populated with previously defined information for the selected event.

   Note: For seeded events with a customization level of Limit, you can only update the event status. For seeded events with a customization level of Core, you cannot update any properties; you can only view the event definition.

   Note: Your Oracle Applications installation may include seeded events owned by Oracle Applications products that you have not licensed. For such events, the Update Event page displays a notice that the event is not licensed. Oracle Workflow will not execute any subscriptions to these events.

2. Enter the internal name of the event in the Name field. The internal name is case-sensitive. The suggested format is a compound structure of identifiers separated by periods (.) as follows:

   <company>.<family>.<product>.<component>.<object>.<event>

3. Enter a display name for the event.
4. Enter an optional description for the event.
5. Select Enabled or Disabled as the event status. If you disable an event, its definition remains in the Event Manager for reference, but you cannot use the event in active subscriptions.
6. If you are defining an event that occurs on your local system, enter a generate function for the event. A generate function is a PL/SQL procedure or Java API that can produce the complete event data from the event name, event key, and an optional parameter list. Define only one generate function for an event, either PL/SQL or Java. See: Standard API for an Event Data Generate Function, page 6-19.
   - To assign the event a PL/SQL generate function, enter the PL/SQL procedure in the Generate Function field in the following format:

     <package_name>.<function_name>

   - To assign the event a Java generate function, enter the Java API in the Java Generate Function field in the following format:

     <customPackage>.<customClass>

7. Identify the program or application that owns the event by entering the program name in the Owner Name field and the program ID in the Owner Tag field.
8. Review the customization level for the event.
   - Core - No changes can be made to the event definition. This level is used only for events seeded by Oracle Applications.
   - Limit - The event status can be updated to Enabled or Disabled, but no other changes can be made to the event definition. This level is used only for events seeded by Oracle Applications.
   - User - Any property in the event definition can be updated. This level is automatically set for events that you define.

To Create or Update an Event Group:

1. Navigate to the Create Group page or to the Update Group page. The Create Group page and the Update Group page are identical, except that the fields in the Update Group page are populated with previously defined information for the selected event group.

   Note: For seeded event groups with a customization level of Limit, you can only update the event group status. For seeded event groups with a customization level of Core, you cannot update any properties; you can only view the event group definition.

   Note: Your Oracle Applications installation may include seeded event groups owned by Oracle Applications products that you have not licensed. For such events, the Update Group page displays a

notice that the event group is not licensed. Oracle Workflow will not execute any subscriptions to these event groups.

2. Enter the internal name of the event group in the Name field. The internal name is case-sensitive. The suggested format is a compound structure of identifiers separated by periods (`.`) as follows:

   `<company>.<family>.<product>.<component>.<object>.<event>`

3. Enter a display name for the event group.
4. Enter an optional description for the event group.
5. Select Enabled or Disabled as the event group status. If you disable an event group, its definition remains in the Event Manager for reference, but you cannot use the event group in active subscriptions.
6. Identify the program or application that owns the event group by entering the program name in the Owner Name field and the program ID in the Owner Tag field.
7. Review the customization level for the event group.
   - Core - No changes can be made to the event group definition. This level is used only for event groups seeded by Oracle Applications.
   - Limit - The event group status can be updated to Enabled or Disabled, but no other changes can be made to the event group definition. This level is used only for event groups seeded by Oracle Applications.
   - User - Any property in the event group definition can be updated. This level is automatically set for event groups that you define.
8. Select the Apply button to save the event group definition.
9. To add a member event to the group, select the Add Events to Group button.

   Note: An event group can contain only individual events as its members. It cannot contain another group.

10. In the Add Events to Group page, search for the events you want to add. The main search option is:
    - Name - Enter the internal name of the event you want to add. You can enter a partial value to search for events whose internal names contain that value.

    You can also enter the following additional search criteria.
    - Display Name - Enter the display name of the event you want to add.
    - Type - You can only add individual events as members of the group.
    - Status - Select Enabled or Disabled as the status of the events you want to add.
    - Owner Name - Enter the name of the program or application that owns the event you want to add.
    - Owner Tag - Enter the program ID of the program or application that owns the event you want to add.
11. Select the event or events that you want to add to your event group, and select the Add to Group button.

12. You can optionally enter new search criteria to search for other events to add to the event group.
13. After you finish adding events to the event group, select the Apply button to save the event group members.
14. To delete a member event from the group, select the event in the Update Group page and select the Delete button.

    Note: Deleting a member event from an event group does not delete the event definition for the individual event. The individual event remains in the Event Manager.

To Raise a Test Event:

1. Navigate to the Test Business Event page.
2. In the Event Identifier region, specify the event you want to raise.
   - Event Name - Select the internal name of the event.

     Note: You cannot raise an event group. You must raise each event individually.

   - Event Key - Enter an event key that uniquely identifies this instance of the event.
   - Send Date - Optionally specify the date when the event message is available for dequeuing. If you set the send date to a future date when you raise the event, the event message is placed on the WF_DEFERRED queue, and subscription processing does not begin until the specified date.
3. In the Event Parameters region, optionally enter any additional parameter name and value pairs to be stored in the parameter list within the event message. You can enter up to 100 parameters.
4. In the Event Data region, you can optionally enter an XML document to describe what occurred in the event. You can enter up to 4000 characters.
   - To enter the event data manually, select the Write XML option and enter the XML document in the XML Content field.
   - To upload an XML file from your file system, select the Upload XML option and select the Upload XML File button. In the Upload XML File page, enter the full path and file name of the XML file containing your event data, and select the Upload File button.

     You can also choose the View link to view the XML file if you have already specified one, or choose the Clear button to remove the previously specified file and enter a different file.

     Note: You can also assign generate functions in the event definition to generate the event data, or raise the event using the *WF_EVENT.Raise()* API instead of using the Test Business Event page. These methods let you provide the event data as a CLOB storing up to four gigabytes of data. See: To Define an Event, page 8-5 and Raise, *Oracle Workflow API Reference*.

Event Subscriptions

An event subscription is a registration indicating that a particular event is significant to a particular system and specifying the processing to perform when the triggering event occurs. You can define your event subscriptions in the Event Manager.

When you install Oracle Workflow, several default subscriptions to predefined Workflow events are automatically created. You can enable, disable, or copy these subscriptions to perform the event processing that you want. See: Predefined Workflow Events, page 9-1.

Whenever an event is raised locally or received from an external source, the Event Manager searches for and executes any eligible subscriptions. To be eligible, a subscription must meet the following requirements:

- The subscriber must be the local system.
- The triggering event must be the event that was raised or received, an event group that includes that event, or the Any event.
- Both the subscription and its triggering event must be marked as active by having a status of Enabled.
- The source type of the subscription must match the source of the event, either local or external.

If no eligible subscriptions exist for the event that occurred (apart from subscriptions to the Any event), then Oracle Workflow executes any enabled subscriptions by the local system to the Unexpected event with the appropriate source type. See: Any Event, page 9-14 and Unexpected Event, page 9-17.

> **Note:** Additionally, for events and subscriptions seeded by Oracle Applications products, both the subscription and its triggering event must be owned by products that are licensed with a status of Shared or Installed, in order for the subscription to be executed.
>
> You can use the License Manager AD utility to review which products you currently have licensed. To ensure that the license status of the seeded events and subscriptions in the Business Event System is updated according to the status of the products you currently have licensed, you can run the Synchronize Product License and Workflow BES License concurrent program. See: License Manager, *Oracle Applications AD Utilies Reference Guide* and Synchronizing License Statuses, *Oracle Workflow Administrator's Guide*.
>
> If you upgrade from an Oracle Applications release earlier than Release 11.5.9, you should run the Synchronize Product License and Workflow BES License concurrent program once after the upgrade to update the license status of the existing events and subscriptions in your Event Manager. Subsequently, when you license a product, Oracle Workflow automatically updates the license status for all the events and subscriptions owned by that product.

Oracle Workflow provides default error handling for subscription processing through a predefined Error subscription to the Unexpected event and the Default Event Error process in the System: Error item type. You can also define custom error handling for your events. See: Error Handling for Event Subscription Processing, page 11-3.

Defining the Subscriber

To begin defining a subscription, you specify which system is the subscriber. The subscriber is the system where you want the subscription to execute.

Each subscription defines an action on exactly one system, so you should define a separate subscription for each system involved in the processing you want to perform. For example, if you want to propagate data from one system to another, you should define one subscription for the sending system and another subscription for the receiving system.

Defining How a Subscription is Triggered

You must specify the source type of the events to which the subscription applies. Events can have the following source types:

- Local - The subscription applies only to events raised on the subscribing system.
- External - The subscription applies only to events received by an inbound agent on the subscribing system.

  Note: All event messages received by an inbound agent on the subscribing system are considered to have an External source, whether the sending agent is located on a remote system or on the local system.

- Error - The subscription applies to only to errored events dequeued from the WF_ERROR queue or the WF_JAVA_ERROR queue.

Next, select the event that you want to trigger the subscription. You can choose either an individual event or an event group. If you choose an event group, the subscription will be triggered whenever any one of the group's member events occurs.

You can also optionally restrict the subscription to be triggered only by events received from a specific source agent. However, in most cases you do not need to specify a source agent.

Controlling How a Subscription is Executed

The phase number for a subscription controls whether the subscription is executed immediately or is deferred. The Event Manager treats subscriptions with a phase number of 100 or higher as deferred subscriptions. Subscriptions with a phase number from 1 to 99 are executed immediately, unless processing for the event is deferred by another method. See: Deferred Subscription Processing, page 8-66.

If you define multiple subscriptions to the same event, the phase numbers for the subscriptions also control the order in which the Event Manager executes those subscriptions. Subscriptions are executed in ascending phase order. For example, you can enter 10 for the subscription that you want to execute first when an event occurs, 20 for the subscription that you want to execute second, and so on. You can use phases to ensure that different types of actions are performed in the appropriate order, such as executing subscriptions that perform validation before subscriptions that perform other types of processing.

Note: If you enter the same phase number for more than one subscription, the Event Manager may execute those subscriptions in any order, relative to each other. However, the Event Manager will still

execute that group of subscriptions in their specified place in the phase order, relative to subscriptions with other phase numbers.

Additionally, you can use phase numbers to ensure that your subscriptions are unique, so that the Workflow XML Loader can properly download and upload the subscription definitions. Each subscription must have a unique combination of subscribing system, source type, triggering event, source agent, phase, owner name, and owner tag. If you are defining several subscriptions with similar properties, assign each subscription a different phase number to differentiate it from the others.

The phase number 0 (zero) is reserved for Oracle Workflow seeded subscriptions.

Depending on the processing to be performed, a subscription may require the complete set of event information contained in the event data, or it may require only the event key that identifies the instance of the event. You can improve performance by specifying Key as the rule data for subscriptions that do not require the complete event data. For locally raised events, the Event Manager checks each subscription before executing it to determine whether the subscription requires the complete event data. If the event data is required but is not already provided, the Event Manager runs the generate function for the event to produce the event data. However, if no subscriptions to the event require the event data, then the Event Manager will not run the generate function, minimizing the resources required to execute the subscriptions.

> **Note:** Even if there are subscriptions that require the complete event data, you can return control to the calling application more quickly after raising the event by deferring all those subscriptions. Then the Event Manager will not run the generate function until those subscriptions are executed at a later time.

Each subscription is assigned a customization level that determines whether you can update the subscription definition. Oracle Workflow uses the customization level to protect Oracle Applications seed data and to preserve your customizations in an upgrade. A subscription can have one of the following customization levels:

- Core - No changes can be made to the subscription definition. This level is used only for subscriptions seeded by Oracle Applications.
- Limit - The subscription status can be updated to Enabled or Disabled, but no other changes can be made to the subscription definition. This level is used only for subscriptions seeded by Oracle Applications.
- User - Any property in the subscription definition can be updated. This level is automatically set for subscriptions that you define.

See: Access Protection for Business Event System Data, *Oracle Workflow Administrator's Guide*.

Some Oracle Applications products provide seeded events and subscriptions. In these cases, Oracle Workflow executes subscriptions only if the triggering event and the subscription are both owned by products that you have licensed with a status of Installed or Shared.

Your Oracle Applications installation may include seeded subscriptions owned by Oracle Applications products that you have not licensed. For such subscriptions, the Update Subscription page displays a notice that the subscription is not licensed. Oracle Workflow will not execute any of these subscriptions. Additionally, Oracle Workflow

will not execute any subscriptions to events that you have not licensed, even if the subscriptions themselves are owned by a product that you have licensed.

You can use the License Manager AD utility to review which products you currently have licensed. To ensure that the license status of the seeded events and subscriptions in the Business Event System is updated according to the status of the products you currently have licensed, you can run the Synchronize Product License and Workflow BES License concurrent program. See: License Manager, *Oracle Applications AD Utilies Reference Guide* and Synchronizing License Statuses, *Oracle Workflow Administrator's Guide*.

> **Note:** Any subscriptions that you define with a customization level of User are always treated as being licensed.

Defining the Action for a Subscription

Subscription processing can include the following types of processing:

- Send the event message to a workflow process.
- Send the event message to an agent.
- Send a notification to a role.
- Receive an Oracle XML Gateway message from your trading partner.
- Send an Oracle XML Gateway message to your trading partner.
- Run a custom function on the event message.

Sending the Event to a Workflow Process

By sending an event to a workflow process, you can model complex processing or routing logic beyond the standard action options. For example, you can branch to different functions, initiate subprocesses, send notifications, or select recipient agents, based on the contents of the event message, or modify the event message itself.

Events are represented within workflow processes by event activities. See: Event Activity, page 3-62.

To send the event to a particular workflow process, you must specify the item type and process name of the process. The item key for the process is determined either by the correlation ID specified in the event message, or by the event key if no correlation ID is specified. The event can either start a new process with that item key or continue an existing process identified by that item key that is waiting to receive it.

> **Note:** The item key for a process instance can only contain single-byte characters. It cannot contain a multibyte value.

> **Note:** You can call *WF_EVENT_FUNCTIONS_PKG.AddCorrelation()* during prior subscription processing to add a correlation ID to the event message. To use *AddCorrelation()*, you must enter a subscription parameter named `ITEMKEY` that specifies a function to generate the correlation ID. The function must be specified in the following format:
>
> `ITEMKEY=<package_name.function_name>`
>
> See: AddCorrelation, *Oracle Workflow API Reference*.

You can also define additional subscription parameters to be used in one of two ways:

- If you want to specify additional parameters to set as item attributes for the workflow process, you can enter these parameters for the subscription and choose the Add Subscription Parameters option to set the subscription parameters into the event message parameter list. The event parameters will then be set as item attributes for the workflow process when the process receives the event.
- If you want to specify additional conditions to control whether or not the subscription is executed, you can enter subscription parameters that describe those conditions and choose the Launch when Parameters Match option. In this case the Event Manager sends the event to the specified workflow process only if the event message includes parameters whose names and values match all the parameters defined for the subscription. Ensure that you include the required parameters in the parameter list within the event message, either by providing them when the event is raised or by adding them during prior subscription processing.

If you want to send the event to multiple existing workflow processes, instead of to one particular process, you can choose the the Launch when Business Key Matches option instead of specifying an item type and process name. In this case, the Event Manager sends the event to all existing workflow process instances that have eligible receive event activities waiting to receive it, marked by a business key attribute that matches the event key. For each receive event activity that you want to receive the event, you must define an activity attribute named `#BUSINESS_KEY`, and set an item type attribute as the default value for that activity attribute. Then include logic to set that item attribute value to match the event key of the corresponding event at runtime.

> **Note:** With this option the event is only sent to continue existing processes. If you want to launch a new process instance with the event, do not select this option.

When an event subscription sends an event to a workflow process, the Workflow Engine performs the following processing:

- Sets any parameters in the event message parameter list as item type attributes for the process, creating new item type attributes if a corresponding attribute does not already exist for any parameter.
- Sets the subscription's globally unique identifier (GUID) as a dynamic item attribute so that the workflow process can reference other information in the subscription definition.
- If the event was originally raised by a Raise event activity in another workflow process, the item type and item key for that process are included in the parameter list within the event message. In this case, the Workflow Engine automatically sets the specified process as the parent for the process that receives the event, overriding any existing parent setting. See: SetItemParent, *Oracle Workflow API Reference*.
- Searches for receive event activities that are eligible to accept the event. For an activity to be eligible, the event must match the activity's event filter, and the activity must either be marked as a Start activity or have a status of '`NOTIFIED`', meaning the process has transitioned to the event.

  > **Note:** If you chose the Launch when Business Key Matches option, then to be eligible an activity must have an event filter that matches the event, a status of '`NOTIFIED`' , and a business key attribute that matches the event key.

- If an event arrives at a Start activity to launch a new process instance, the Workflow Engine also searches for all other receive event activities that are marked as Start activities and that do not have any incoming transitions, regardless of their event filter. For these activities, the Workflow Engine sets the activity status to 'NOTIFIED' so that they will be ready to receive an event if any more events are sent to this process. This feature lets you design a workflow process that requires multiple events to be received when you do not know in advance the order in which the events will arrive.
- Stores the event name, event key, and event message in item type attributes, as specified in each eligible activity node's event details.
- Marks all the eligible event activity nodes with a status of 'COMPLETED' and continues the thread of execution from each of those nodes.

See: Default_Rule, *Oracle Workflow API Reference*, Default_Rule2, *Oracle Workflow API Reference*, Default_Rule3, *Oracle Workflow API Reference*, and Instance_Default_Rule, *Oracle Workflow API Reference*.

> **Note:** You can also send an event to a workflow process by choosing the custom action option and including send processing in your custom rule function. See: Standard API for an Event Subscription Rule Function, page 6-23.

Sending the Event to an Agent

To send an event to an agent, you must specify either the Out Agent that should send the outbound message, or the To Agent that should receive the inbound message, or both.

- If you specify both a To Agent and an Out Agent, Oracle Workflow places the event message on the Out Agent's queue for propagation, addressed to the To Agent.
- If you specify a To Agent without an Out Agent, Oracle Workflow selects an outbound agent on the subscribing system whose queue type matches the queue type of the To Agent. The event message is then placed on this outbound agent's queue for propagation, addressed to the To Agent.
- If you specify an Out Agent without a To Agent, Oracle Workflow places the event message on the Out Agent's queue without a specified recipient.
  - You can omit the To Agent if the Out Agent uses a multi-consumer queue with a subscriber list. (The standard Workflow queue handlers work only with multi-consumer queues.) In this case the queue's subscriber list determines which consumers can dequeue the message. If no subscriber list is defined for that queue, however, the event message is placed on the WF_ERROR queue for error handling.

    > **Note:** The subscriber list for a multi-consumer queue in Oracle Advanced Queuing is different from event subscriptions in the Oracle Workflow Business Event System. For more information, see: Subscription and Recipient Lists, *Oracle Application Developer's Guide - Advanced Queuing* or *Oracle Streams Advanced Queuing User's Guide and Reference*.

  - You can also omit the To Agent if the Out Agent uses a single-consumer queue for which you have defined a custom queue handler. For a single-consumer queue, no specified consumer is required.

You can optionally specify the priority with which the recipient should dequeue a message. Messages are dequeued in ascending priority order.

You can also define additional subscription parameters to be used in one of two ways:

- If you want to add more parameters to the event message before it is sent, you can enter these parameters for the subscription and choose the Add Subscription Parameters option to set the subscription parameters into the event message parameter list.
- If you want to specify additional conditions to control whether or not the subscription is executed, you can enter subscription parameters that describe those conditions and choose the Launch when Parameters Match option. In this case the Event Manager sends the event to the specified agent only if the event message includes parameters whose names and values match all the parameters defined for the subscription. Ensure that you include the required parameters in the parameter list within the event message, either by providing them when the event is raised or by adding them during prior subscription processing.

If you want an event message to become available to the recipient at a future date, rather than being available immediately as soon as it is propagated, you can set the `SEND_DATE` attribute within the event message to the date you want. You should set the send date during prior subscription processing before the event is sent. The event message is propagated to the To Agent but does not become available for dequeuing until the specified date.

See: Default_Rule, *Oracle Workflow API Reference*, Default_Rule2, *Oracle Workflow API Reference*, and Default_Rule3, *Oracle Workflow API Reference*.

> **Note:** You can also send an event to an agent by choosing the custom action option and including send processing in your custom rule function. See: Standard API for an Event Subscription Rule Function, page 6-23.

Sending a Notification

If the only processing you need to perform when an event occurs is sending a notification, you can define a subscription that simply performs that action. In this case you do not need to define and run a complete workflow process to send the notification.

However, you must specify a message template for the notification you want to send. The message template must be a message defined within an item type in the Workflow Builder and stored in your database. You can either select a standard message provided by Oracle Applications or define a custom message.

You must also specify the role that should receive the notification. The notification will appear in the recipient's worklist.

You can optionally specify a custom callback function you want the Notification System to use for communication of SEND and RESPOND source message attributes. Otherwise, Oracle Workflow uses a standard default callback function. See: Custom Callback Function, *Oracle Workflow API Reference*.

You can also optionally enter context information that you want to pass to the callback function. With the standard Oracle Workflow callback function, the Notification System requires a context to obtain values for item type attributes. If you do not specify a context, do not reference any item type attributes in the message attributes. Otherwise

the message body will not display correctly. The context information consists of the item type, item key, and activity ID in the following format:

`<itemtype>:<itemkey>:<activityid>`

If you specify a custom callback function, then you can enter the context information that is appropriate for your function.

You can also optionally enter a comment to include with the message, a due date, and a priority for the message.

After sending the notification, the Event Manager sets the notification ID into the event parameter list as a parameter named `#NID`. If you want to use the notification ID in further processing, raise the event using *WF_EVENT.Raise3()*, which returns the event parameter list after the Event Manager completes subscription processing for the event. You can then call *WF_EVENT.GetValueForParameter()* to obtain the value of the `#NID` parameter. See: Raise3, *Oracle Workflow API Reference* and GetValueForParameter, *Oracle Workflow API Reference*.

For example, if the notification requires a response, you can retrieve the response values from the user's reply by obtaining the notification ID and using it to call *WF_NOTIFICATION.GetAttrText()*, *WF_NOTIFICATION.GetAttrNumber()*, or *WF_NOTIFICATION.GetAttrDate()* for the RESPOND attributes. See: GetAttribute, *Oracle Workflow API Reference*.

See: SendNotification, *Oracle Workflow API Reference*.

Receiving and Sending Oracle XML Gateway Messages

If you use Oracle XML Gateway, you can define subscriptions either to receive an Oracle XML Gateway message from your trading partner, or to generate an Oracle XML Gateway message and send the message to your trading partner. Oracle Workflow and Oracle XML Gateway provide these actions to support business-to-business (B2B) integration scenarios.

The event that triggers such a subscription must include the trading partner information within its event parameter list. Oracle XML Gateway then uses a standard workflow process defined in the XML Gateway Standard item type to send or receive the message.

You can optionally specify the priority with which the recipient should dequeue a message. Messages are dequeued in ascending priority order.

See: XML Gateway Standard Item Type, *Oracle XML Gateway User's Guide*.

Running a Custom Rule Function

You can extend your subscription processing by creating custom rule functions. A subscription can have either a PL/SQL rule function for processing in the database or a Java rule function for processing in the middle tier.

Custom rule functions must be defined according to a standard PL/SQL or Java API. See: Standard API for an Event Subscription Rule Function, page 6-23.

You can use a rule function for many different purposes, including:

- Performing validation
- Processing inbound messages as a receive message handler for an application
- Making modifications to an outbound message, such as adding a correlation ID that associates this message with other messages

> **Note:** You can call *WF_EVENT_FUNCTIONS_PKG.AddCorrelation()* within a custom rule function to add a correlation ID during your custom processing. See: AddCorrelation, *Oracle Workflow API Reference*.

A rule function may read or write to the event message or perform any other database action. However, you should never commit within a rule function. The Event Manager never issues a commit as it is the responsibility of the calling application to commit. Additionally, the function must not change the connection context in any way, including security and NLS settings. If a rule function returns an error, subscription processing is halted.

To run a function on the event message, you must specify the rule function that you want to execute. You can also specify any additional parameters that you want to pass to the function.

If you want to send the event message to a workflow process or to an agent as part of your custom processing, you can specify the workflow item type, process name, Out Agent, and To Agent in the subscription definition, for your rule function to reference. Note, however, that you must explicitly include the send processing in your rule function.

You can optionally specify the priority with which the event message should be dequeued if it is placed on an agent, such as a standard agent for deferred subscription processing. Messages are dequeued in ascending priority order.

Oracle Workflow provides standard rule functions that perform subscription processing for the standard subscription actions. You can also use these rule functions in a custom subscription if you choose. Oracle Workflow also provides some standard rule functions that you can use for testing and debugging or other purposes. See: Event Subscription Rule APIs, *Oracle Workflow API Reference*.

If the subscription processing that you want to perform for an event includes several successive steps, you may find it advantageous to define multiple subscriptions to the event with simple rule functions that you can reuse, rather than creating complex specialized rule functions that cannot be reused. You can enter phase values for the subscriptions to specify the order in which they should be executed.

You can run a diagnostic test through Oracle Diagnostics to verify that the rule functions that are defined for event subscriptions exist in the database and are valid. See: Oracle Workflow Diagnostic Tests, *Oracle Workflow Administrator's Guide*.

Documenting Identifying Information for a Subscription

You can associate a subscription with the program or application to which it belongs by setting the program name and brief identifier as the owner name and owner tag for the subscription. The program can then use this identifying information to locate the subscriptions that it owns. The subscription owner may be the same program as the owner of the triggering event, or a different program.

Deferred Subscription Processing

If you do not want subscriptions for an event to be executed immediately when the event occurs, you can defer the subscriptions. In this way you can return control more quickly

to the calling application and let the Event Manager execute any costly subscription processing at a later time.

You can defer subscription processing by three different methods:

- Raise the event with a future date in the `SEND_DATE` attribute. Use this method when you want to defer all subscription processing for a locally raised event until a particular effective date.
- Define subscriptions to the event with phase numbers of 100 or higher. Use this method when you want to defer processing of particular subscriptions for either local or external events.
- Set the dispatch mode of the Event Manager to deferred processing before raising the event from PL/SQL code. This method can be used to defer all subscription processing for a locally raised event. This method is not recommended, however, and should only be used in exceptional circumstances.

  Note: Additionally, if an event is raised by PL/SQL code or received by a PL/SQL agent listener, the Event Manager always defers processing of any Java subscriptions to the event, regardless of their phase numbers.

When subscription processing for an event is deferred by any of these methods, the event message is placed on a standard deferred agent. If the subscription at which processing is deferred is a PL/SQL subscription, the event is placed on the standard WF_DEFERRED agent. If the subscription at which processing is deferred is a Java subscription, the event is placed on the standard WF_JAVA_DEFERRED agent. You must ensure that agent listeners are running to monitor the WF_DEFERRED and WF_JAVA_DEFERRED agents. See: Scheduling Listeners for Local Inbound Agents, *Oracle Workflow Administrator's Guide*.

The listener dequeues event messages from the WF_DEFERRED agent or WF_JAVA_DEFERRED agent in priority order. The event messages retain their original source type, whether Local or External. The amount of time by which subscription processing for these events is deferred depends on the schedule defined for the listener, and, for future-dated events, on the specified effective date.

Note: If an event was deferred when the Event Manager encountered a Java subscription during subscription processing in the database, the Event Manager executes all remaining subscriptions to the event when the event is dequeued from the WF_JAVA_DEFERRED queue, including both Java and PL/SQL subscriptions, regardless of the subscription phase numbers. Subscriptions will not be deferred a second time.

Deferring Subscription Processing Using a Future Send Date

You can defer subscription processing for a local event until a particular future effective date by raising the event with that date in the `SEND_DATE` attribute. For example, you could enter information for a new employee in a human resources application as soon as the employee was hired, but defer payroll processing until the employee's start date.

When an event is raised with a future send date, the Event Manager immediately places the event message on the deferred queue, without executing any of the subscriptions for the event. All subscriptions to the event are deferred, regardless of their phase number. The event remains in a `WAIT` state until the send date. When the send date arrives, the event message becomes available for dequeuing and will be dequeued the next time an agent listener runs on the deferred queue. The amount of time by which

subscription processing is deferred depends on the send date you specify as well as on the schedule defined for the listener.

When the listener dequeues the event message, the Event Manager checks for a subscription ID in the `ERROR_SUBSCRIPTION` attribute. If the event message does not contain a subscription ID, meaning that all subscription processing for the event was deferred immediately after the event was raised, then the Event Manager proceeds to execute all subscriptions to the event, in ascending phase order.

> **Note:** If an event was deferred when it was raised, the Event Manager executes all eligible subscriptions to the event when the event is dequeued from the deferred queue, regardless of the subscription phase numbers. Subscriptions will not be deferred a second time, even if they have a phase number of 100 or higher.

See: Raise, *Oracle Workflow API Reference*.

Deferring Subscription Processing Using Subscription Phase Numbers

You can also use the phase number for a subscription to control whether the subscription is executed immediately or is deferred. The Event Manager treats subscriptions with a phase number of 100 or higher as deferred subscriptions. Both Local and External subscriptions can be deferred in this way.

> **Note:** For this deferral method to take effect when an event is raised locally, the event must not be raised with a future send date, and the Event Manager must be in the normal synchronous mode for subscription processing. Otherwise, the event message will immediately be placed on the deferred queue, and the Event Manager will not execute any subscriptions until the event is dequeued from there.

When a triggering event is raised or received, the Event Manager executes subscriptions to that event in phase order until it encounters a subscription with a phase number of 100 or higher. The Event Manager sets that subscription into the `ERROR_SUBSCRIPTION` attribute within the event message, as well as setting the priority specified in the subscription properties into the `PRIORITY` attribute. Then the event message is placed on the appropriate deferred queue.

> **Note:** Additionally, if an event is raised by PL/SQL code or received by a PL/SQL agent listener, and the Event Manager encounters a Java subscription to the event, it always defers the remaining subscription processing beginning with that subscription, regardless of the subscription phase number.

The amount of time by which subscription processing is deferred depends on the schedule defined for the agent listener monitoring the WF_DEFERRED or WF_JAVA_DEFERRED agent. When the listener dequeues an event message, the Event Manager checks for a subscription ID in the ERROR_SUBSCRIPTION attribute. If a subscription ID is present, meaning that subscription processing was deferred from that subscription onwards, the Event Manager begins by executing that subscription, and then continues executing any other subscriptions to the event with the same or a higher phase number.

> **Note:** The Event Manager resumes subscription processing at the phase number of the subscription set into the event message. It does not necessarily begin with the phase number 100, if there were no subscriptions with that phase number when the event was originally processed.

Deferring Subscription Processing Using the Event Manager Dispatch Mode

If you raise an event from a local application in PL/SQL code, you can also choose to defer all subscription processing for that event every single time the application raises it. To do so, call the *SetDispatchMode()* API with the mode '`ASYNC`', indicating deferred (asynchronous) processing, just before calling the *Raise()* API. See: SetDispatchMode, *Oracle Workflow API Reference*.

This method is not recommended, however, and should only be used in exceptional circumstances, since it requires hard-coding the deferral in your application. To retain the flexibility to modify subscription processing without intrusion into the application, you can simply raise the event with a future send date or mark some or all of the individual subscriptions for deferral using the subscription phase numbers.

When an event is raised after the dispatch mode is set to deferred processing, the Event Manager immediately places the event message on the WF_DEFERRED queue, without executing any of the subscriptions for the event. All subscriptions to the event are deferred, regardless of their phase number.

The amount of time by which subscription processing is deferred depends on the schedule defined for the agent listener monitoring the WF_DEFERRED agent. When the listener dequeues the event message, the Event Manager checks for a subscription ID in the `ERROR_SUBSCRIPTION` attribute. If the event message does not contain a subscription ID, meaning that all subscription processing for the event was deferred immediately after the event was raised, then the Event Manager proceeds to execute all subscriptions to the event, in ascending phase order.

> **Note:** If an event was deferred when it was raised, the Event Manager executes all eligible subscriptions to the event when the event is dequeued from the WF_DEFERRED queue, regardless of the subscription phase numbers. Subscriptions will not be deferred a second time, even if they have a phase number of 100 or higher.

Subscription Processing for PL/SQL and Java Subscriptions

If an event is raised locally from PL/SQL code, or received on an agent and processed by a PL/SQL agent listener, the Event Manager performs subscription processing in the database tier. It begins processing the subscriptions to the event in phase order and executes PL/SQL subscriptions synchronously, within the same database session, as long as the subscriptions are not deferred.

If the Event Manager encounters a PL/SQL subscription that is deferred, it stops processing and places the event message on the WF_DEFERRED agent. Subscription processing for the event resumes when the Workflow Deferred Agent Listener or another agent listener runs on that agent.

If the Event Manager encounters any Java subscription, or if the event has a Java generate function, the Event Manager stops processing and places the event message on the WF_JAVA_DEFERRED agent, regardless of the subscription phase number. Subscription

processing for the event resumes when the Workflow Java Deferred Agent Listener or another agent listener runs on that agent.

If an event is raised locally from Java code, or received on an agent and processed by a Java agent listener, the Event Manager performs subscription processing in the middle tier. It begins processing the subscriptions to the event in phase order, executing Java subscriptions in Java and PL/SQL subscriptions using Java Database Connectivity (JDBC), as long as the subscriptions are not deferred.

If the Event Manager encounters a Java subscription that is deferred, it stops processing and places the event message on the WF_JAVA_DEFERRED agent. Subscription processing for the event resumes when the Workflow Java Deferred Agent Listener or another agent listener runs on that agent.

If the Event Manager encounters a PL/SQL subscription that is deferred, it stops processing and places the event message on the WF_DEFERRED agent. Subscription processing for the event resumes when the Workflow Deferred Agent Listener or another agent listener runs on that agent.

Related Topics

To View and Maintain Event Subscriptions, page 8-70

To Create or Update an Event Subscription, page 8-71

Standard API for an Event Subscription Rule Function, page 6-23

Scheduling Listeners for Local Inbound Agents, *Oracle Workflow Administrator's Guide*.

Defining Event Subscriptions

To View and Maintain Event Subscriptions:

1. Use a Web browser to navigate to the Event Manager, using a responsibility and navigation path specified by your system administrator. Then choose Subscriptions in the horizontal navigation. See: Oracle Workflow Developer Navigation Paths, page A-1.
2. Search for the subscriptions you want to display. The main search option is:
   - System - Select the system for which you want to display subscriptions.

   You can also enter the following additional search criteria.
   - Source Type - Select Local, External, or Error as the type of source system for which you want to display subscriptions.
   - Event - Select the event for which you want to display subscriptions.
   - Status - Select Enabled or Disabled as the status of the subscriptions you want to display.
3. If you have workflow administrator privileges, you can use the administration icons and buttons to perform administrative operations.
   - To update a subscription, choose the update icon for that subscription. See: To Create or Update an Event Subscription, page 8-71.

     Note: For seeded subscriptions with a customization level of Limit, you can only update the subscription status. For seeded

subscriptions with a customization level of Core, you cannot update any properties; you can only view the subscription definition.

- To delete a subscription, select the subscription and select the Delete button.

  Note: You cannot delete any subscription seeded by Oracle Applications that has a customization level of Core or Limit.

- To create a new subscription, select the Create Subscription button. See: To Create or Update an Event Subscription, page 8-71.

To Create or Update an Event Subscription:

1. Navigate to the Create Event Subscription page or to the Update Event Subscriptions page. The Create Event Subscription page and the Update Event Subscriptions page are identical, except that the fields in the Update Event Subscriptions page are populated with previously defined information for the selected subscription.

   Note: For seeded subscriptions with a customization level of Limit, you can only update the subscription status. For seeded subscriptions with a customization level of Core, you cannot update any properties; you can only view the subscription definition.

   Note: Your Oracle Applications installation may include seeded subscriptions owned by Oracle Applications products that you have not licensed. For such subscriptions, the Update Subscription page displays a notice that the subscription is not licensed. Oracle Workflow will not execute any of these subscriptions.

2. In the Subscriber region, select the system where the subscription executes.
3. In the Triggering Event region, specify the event source to which the subscription applies in the Source Type field.
   - Local - The subscription applies only to events raised on the subscribing system.
   - External - The subscription applies only to events received by an inbound agent on the subscribing system.

     Note: All event messages received by an inbound agent on the subscribing system are considered to have an External source, whether the sending agent is located on a remote system or on the local system.

   - Error - The subscription applies to only to errored events dequeued from the WF_ERROR queue or WF_JAVA_ERROR queue.
4. Select the event that triggers the subscription in the Event Filter field. You can specify an individual event or an event group.
5. Optionally select a source agent to which the subscription applies. If you specify a source agent, then the subscription is executed only when the triggering event is received from that agent.

Note: In most cases, the Source Agent field is left blank.

6. In the Execution Condition region, enter a phase number for the subscription to specify the order in which subscriptions triggered by the same event are executed. The phase number also controls whether a subscription is executed immediately or is deferred, unless processing for the event is deferred by another method.

   Note: Each subscription must have a unique combination of subscribing system, source type, triggering event, source agent, phase, owner name, and owner tag. If you are defining several subscriptions with similar properties, assign each subscription a different phase number to differentiate it from the others.

7. Select Enabled or Disabled as the subscription status. If you disable a subscription, it still remains in the Event Manager for reference, but it can no longer be executed when events occur.
8. In the Rule Data field, specify the event information required by the subscription.
   - Key - The subscription requires only the event key.
   - Message - The subscription requires the complete event data.
9. In the Action Type region, select the subscription processing you want to perform when the triggering event occurs.
   - Launch Workflow - Send the event message to launch or continue a workflow process.
   - Send to Agent - Send the event message to a Business Event System agent.
   - Send Notification - Send a notification using a standard or custom message template.
   - Receive Trading Partner Message - Receive an Oracle XML Gateway message from your trading partner.
   - Send Trading Partner Message - Generate an Oracle XML Gateway message and send the message to your trading partner.
   - Custom - Execute custom business logic defined as a PL/SQL rule function or as a Java rule function.
10. Specify the error handling to perform if Oracle Workflow encounters an error while processing this subscription. Currently, Oracle Workflow supports Stop and Rollback error handling, in which the Event Manager halts all subscription processing for the event, rolls back any subscriptions already executed for the event, and places the event message on the standard WF_ERROR or WF_JAVA_ERROR agent.

    Note: The Skip to Next option is reserved for future use.

    See: Error Handling for Event Subscription Processing, page 11-3.

11. Choose the Next button to define the details of the subscription action, depending on the action type you selected.
12. If you selected Launch Workflow as the action type, enter the following details.

- Select the workflow item type and process name for the workflow process to which you want to send the event. Ensure that the process contains a receive event activity that will be able to accept the event. The event can either launch a new process instance or continue an existing process instance that is waiting to receive the event.

  Note: The list of values for the Workflow Process field includes only the runnable processes within the item type you specify.

- Select Normal, High, or Low as the priority for the subscription. Oracle Workflow uses the priority to help determine the order in which deferred subscriptions are processed. The default priority is Normal.
- You can optionally select an additional option to control how the subscription is executed.
  - Add Subscription Parameters - Set the subscription parameters into the parameter list within the event message before sending the event to the specified workflow process. All the event parameters will be set as item attributes for the workflow process when the process receives the event.
  - Launch when Business Key Matches - Send the event to all existing workflow process instances that have eligible receive event activities waiting to receive it, marked by a business key attribute that matches the event key. This option lets you send the event to multiple processes. However, note that with this option the event is only sent to continue existing processes. If you want to launch a new process instance with the event, do not select this option.

    Note: If you select this option, the Event Manager ignores the specified workflow item type and process name. Instead, it uses the business key attribute to identify the workflow processes that should receive the event.

  - Launch when Parameters Match - Send the event to the specified workflow process only if the event message includes parameters whose names and values match all the parameters defined for the subscription. This option lets you specify additional conditions to control whether or not the subscription is executed. Ensure that you include the required parameters in the parameter list within the event message, either by providing them when the event is raised or by adding them during prior subscription processing.
- If you selected the Add Subscription Parameters option or the Launch when Parameters Match option, specify the parameters for the subscription. Enter the name and value for each parameter with no spaces.

13. If you selected Send to Agent as the action type, enter the following details.
    - Enter either the Out Agent that you want to send the outbound message, or the To Agent that you want to receive the inbound message, or both.
      - If you specify both a To Agent and an Out Agent, Oracle Workflow places the event message on the Out Agent's queue for propagation, addressed to the To Agent.
      - If you specify a To Agent without an Out Agent, Oracle Workflow selects an outbound agent on the subscribing system whose queue type matches

the queue type of the To Agent. The event message is then placed on this outbound agent's queue for propagation, addressed to the To Agent.

- If you specify an Out Agent without a To Agent, Oracle Workflow places the event message on the Out Agent's queue without a specified recipient. The Out Agent must use either a multi-consumer queue with a subscriber list or a single-consumer queue.

  Note: The Out Agent must be located on the subscribing system. The list of values for the Out Agent field includes only agents with a direction of Out.

  The list of values for the To Agent field includes only agents with a direction of In.

- Select Normal, High, or Low as the priority for the subscription. Oracle Workflow uses the priority to help determine the order in which deferred subscriptions are processed. The default priority is Normal.
- You can optionally select an additional option to control how the subscription is executed.
  - Add Subscription Parameters - Set the subscription parameters into the parameter list within the event message before placing the event message on the outbound agent.
  - Launch when Parameters Match - Place the event message on the outbound agent only if the event message includes parameters whose names and values match all the parameters defined for the subscription. This option lets you specify additional conditions to control whether or not the subscription is executed. Ensure that you include the required parameters in the parameter list within the event message, either by providing them when the event is raised or by adding them during prior subscription processing.
- If you selected the Add Subscription Parameters option or the Launch when Parameters Match option, specify the parameters for the subscription. Enter the name and value for each parameter with no spaces.

14. If you selected Send Notification as the action type, enter the following details.
    - Select the workflow item type and message name for the message you want to send.

      Note: You do not need to define or run a workflow process to send a notification from a subscription with the Send Notification action type. However, you do need to define the message you want to send within a workflow item type. The list of values for the Message Name field includes only the messages within the item type you specify.

    - Select the role that should receive the notification.
    - Optionally enter a custom callback function you want the Notification System to use for communication of SEND and RESPOND source message attributes. See: Custom Callback Function, *Oracle Workflow API Reference*.
    - Optionally enter context information for a workflow process instance that you want to pass to the callback function. With the standard Oracle Workflow

callback function, the Notification System requires a context to obtain values for item type attributes. If you do not specify a context, do not reference any item type attributes in the message attributes. Otherwise the message body will not display correctly. The context information consists of the item type, item key, and activity ID in the following format:

```
<itemtype>:<itemkey>:<activityid>
```

If you specify a custom callback function, then enter the context information that is appropriate for your function.

- Optionally enter a comment to send with the message.
- Optionally enter a due date when a response is required. This date is only for the recipient's information; it has no effect on processing.
- Select Normal, High, or Low as the priority for the message. Oracle Workflow also uses the priority to help determine the order in which deferred subscriptions are processed. The default priority is Normal.

15. If you selected Receive Trading Partner Message or Send Trading Partner Message as the action type, select Normal, High, or Low as the priority for the subscription. Oracle Workflow uses the priority to help determine the order in which deferred subscriptions are processed. The default priority is Normal.

    Note: You do not need to enter any additional subscription parameters for these action types.

16. If you selected Custom as the action type, enter the following details.
    - Enter the rule function that contains the custom business logic you want to run on the event message. You can define the rule function either as a Java class or as a PL/SQL function. In either case, the rule function must be defined according to a standard API. See: Standard API for an Event Subscription Rule Function, page 6-23.
      - For a Java subscription, enter the Java rule function in the following format:

        ```
        <customPackage>.<customClass>
        ```

        In this case, you should leave the PL/SQL Rule Function field blank.
      - For a PL/SQL subscription, enter the PL/SQL rule function in the following format:

        ```
        <package_name>.<function_name>
        ```

        In this case, you should leave the Java Rule Function field blank.

        Note: If you enter a rule function other than WF_RULE. Default_Rule, Oracle Workflow does not automatically send the event message to the specified workflow and agent. You must explicitly include the send processing in your custom rule function instead.

    - If your rule function will send the event to a workflow process, you can specify the workflow item type and process name for the function to reference. Ensure that the process contains a receive event activity that will be able to accept the event.

Note: The list of values for the Workflow Process field includes only the runnable processes within the item type you specify.

- If your rule function will send the event to an agent, you can specify the outbound and inbound agents for the function to reference. Depending on the send processing in your function, you may need to specify either the Out Agent that you want to send the outbound message, or the To Agent that you want to receive the inbound message, or both.

  Note: The Out Agent must be located on the subscribing system. The list of values for the Out Agent field includes only agents with a direction of Out.

  The list of values for the To Agent field includes only agents with a direction of In.

- Select Normal, High, or Low as the priority for the subscription. Oracle Workflow uses the priority to help determine the order in which deferred subscriptions are processed. The default priority is Normal.
- You can optionally specify additional parameters for the subscription, to be referenced by the rule function. Enter the name and value for each parameter with no spaces.

  Oracle Workflow converts this list of subscription parameters into a text string of name and value pairs, separated by spaces, in the following format:

  <name1>=<value1> <name2>=<value2> ... <nameN>=<valueN>

  The rule function can reference the parameters from this format.

17. For all action types, in the Documentation region, identify the program or application that owns the subscription by entering the program name in the Owner Name field and the program ID in the Owner Tag field.
18. Review the customization level for the subscription.
    - Core - No changes can be made to the subscription definition. This level is used only for subscriptions seeded by Oracle Applications.
    - Limit - The subscription status can be updated to Enabled or Disabled, but no other changes can be made to the subscription definition. This level is used only for subscriptions seeded by Oracle Applications.
    - User - Any property in the subscription definition can be updated. This level is automatically set for subscriptions that you define.
19. Enter an optional description for the subscription.

Agents

An agent is a named point of communication within a system. Communication within and between systems is accomplished by sending a message from one agent to another. A single system can have several different agents representing different communication alternatives. For example, a system may have different agents to support inbound and outbound communication, communication by different protocols, different propagation frequencies, or other alternatives.

You should define each agent that you will use to communicate events in the Event Manager. Each agent's name must be unique within its system. The agent can be referenced in code within Oracle Workflow by a compound name in the following format:

```
<agent_name>@<system_name>
```

For example, the agent WF_IN within the system HUB could be referenced as `WF_IN@HUB`.

After defining the agents on your local system, you should set them up for event message communication by scheduling agent listeners for local inbound agents and propagation for local outbound agents. You can use Oracle Enterprise Manager to schedule propagation and the Workflow Manager component of Oracle Applications Manager to run agent listeners. In Oracle Workflow Manager you can also view the distribution of event messages on different agents, drill down to view details about individual event messages, and review queue details for the agents. For more information, please refer to the Oracle Applications Manager online help, Oracle Enterprise Manager online help, Oracle Enterprise Manager Support, *Oracle Application Developer's Guide - Advanced Queuing* or *Oracle Streams Advanced Queuing User's Guide and Reference*, Distributed Management, *Oracle Enterprise Manager Administrator's Guide*, and Setting Up the Business Event System, *Oracle Workflow Administrator's Guide*.

You can run a diagnostic test through Oracle Diagnostics to verify the status of your agents. See: Oracle Workflow Diagnostic Tests, *Oracle Workflow Administrator's Guide*.

Assigning a Direction to an Agent

When you define an agent in the Event Manager, you must specify the direction of communication that the agent supports on its local system.

- In - Inbound communication to the system. The agent receives messages in a specific protocol and presents them to the system in a standard format.
- Out - Outbound communication from the system. The agent accepts messages from the system in a standard format and sends them using the specified protocol.

Assigning a Protocol to an Agent

You must associate each agent with the protocol by which it communicates messages. The protocol specifies how the messages are encoded and transmitted. For a message to be successfully communicated, the sending and receiving agents must use the same protocol.

A protocol can represent a network standard, such as SQLNET. It can also represent a business-to-business standard that defines the higher-level message format and handshaking agreements between systems in addition to the network standard.

The Business Event System interacts with an agent through an AQ queue. You can use AQ to perform the propagation of messages by the SQLNET protocol which it supports. In Oracle9*i* Database and higher, AQ also includes Internet access functionality that lets you perform AQ operations over the Internet by using AQ's Internet Data Access Presentation (IDAP) for messages and transmitting the messages over the Internet using transport protocols such as HTTP or HTTPS. Additionally, the Messaging Gateway feature of AQ in Oracle9*i* Database and higher enables communication between applications based on non-Oracle messaging systems and AQ, letting you integrate with third party messaging solutions. You can also implement other services to propagate messages by different protocols.

To implement a custom protocol, you must perform the following steps:

1. Define AQ queues to hold pending inbound and outbound messages.
2. Provide code that propagates messages to and from the AQ queues.
3. Define a lookup code for the new protocol in the Event Protocol Type (`WF_AQ_PROTOCOLS`) lookup type, which is stored in the Standard item type. The Event Manager uses the Event Protocol Type lookup type to validate the protocol for an agent. See: To Create Lookup Codes for a Lookup Type, page 3-21.
4. Define agents with the new protocol. See: To Create or Update an Agent, page 8-89.

If an agent supports inbound communication, you must specify the address by which systems can communicate with it. The format of the address depends on the agent's protocol. For agents that use the SQLNET protocol, the address must be in the following format to enable AQ propagation:

```
<schema>.<queue>@<database link>
```

In this format, *`<schema>`* represents the schema that owns the queue, *`<queue>`* represents the queue name, and *`<database link>`* represents the name of the database link to the instance where the queue is located.

> **Note:** You must enter the database link name exactly as the name was specified when the database link was created. For example, if a database link is named `ORA10.US.ORACLE.COM`, you must enter that complete name in the address of an agent on that database. You cannot abbreviate the name to `ORA10`.
>
> The names of the database links that you want to use for the Business Event System should be fully qualified with the domain names. To confirm the names of your database links, use the following syntax:
>
> ```
> SELECT db_link FROM all_db_links
> ```
>
> See: Creating Database Links, *Oracle Workflow Administrator's Guide*.

Assigning a Queue to an Agent

You must associate each agent on a Workflow-enabled system with an AQ queue. The local system uses this queue to interact with the agent. To send messages, the system enqueues the messages on the queue and sets the recipient addresses. To receive messages, the system runs a listener on the queue. See: Setting Up Queues, *Oracle Workflow Administrator's Guide*.

Event messages within the Oracle Workflow Business Event System are encoded in a standard format defined by the datatype `WF_EVENT_T`, or, in Java, the `BusinessEvent` object. You should assign each agent a PL/SQL or Java package called a queue handler that translates between the standard Workflow format and the format required by the agent's queue. Define only one queue handler for an agent, either PL/SQL or Java. See: Event Message Structure, *Oracle Workflow API Reference*.

> **Note:** Even if the agent's queue uses `WF_EVENT_T` as its payload type, a queue handler is still required in order to set native AQ message properties.

Oracle Workflow provides two standard queue handlers, called WF_EVENT_QH and WF_ERROR_QH, for queues that use SQLNET propagation and use the `WF_EVENT_T` datatype as their payload type. You can use WF_EVENT_QH with queues that handle normal Business Event System processing, while WF_ERROR_QH should be used exclusively with error queues.

Oracle Workflow also provides a standard queue handler called WF_EVENT_OJMSTEXT_QH for queues that use the `SYS.AQ$_JMS_TEXT_MESSAGE` datatype as their payload type. This queue handler enables communication of JMS Text messages through the Business Event System. See: Mapping Between WF_EVENT_T and SYS.AQ$_JMS_TEXT_MESSAGE, *Oracle Workflow API Reference*.

If you want to use queues that require a different format, create a custom queue handler for that format. Your custom queue handler must include a set of standard APIs to enqueue and dequeue messages in the custom format. See: Standard APIs for a Queue Handler, page 6-21.

- If you define a PL/SQL queue handler, then you can run both PL/SQL and Java agent listeners on this agent. The PL/SQL queue handler is used during both Java and PL/SQL event message processing. Java agent listeners will execute the PL/SQL queue handler through Java Database Connectivity (JDBC).
- If you define a Java queue handler, then you can run only Java agent listeners on this agent. The Java queue handler is used during Java event message processing. You cannot run a PL/SQL agent listener on this agent, because a PL/SQL agent listener cannot execute a Java queue handler.

Agent Groups

You can also define agent groups that let you associate several inbound agents with each other and reference them as a group in event subscriptions and Send event activities. An agent group is a type of agent composed of a set of individual member agents. Once you have defined an agent group, you can send event messages to the group rather than having to send the messages separately to each individual agent within it.

Agent groups can only be used for inbound communication. All agent groups have a direction of In, and only individual agents with a direction of In can be members of an agent group.

You must associate each agent group with a system to which it belongs. However, you can include agents on other systems within the group.

Ensure that you run an agent listener for each agent within the group to receive inbound messages.

> **Note:** You cannot run an agent listener for an agent group. Agent listeners can be run only for individual agents.

Standard Agents

When you install Oracle Workflow, several standard agents are automatically defined for the Business Event System.

- WF_CONTROL - Oracle Workflow internal agent, not for customer use
- WF_DEFERRED - Standard agent for deferred subscription processing in the database

- WF_ERROR - Standard agent for error handling in the database
- WF_JAVA_DEFERRED - Standard agent for deferred subscription processing in the middle tier
- WF_JAVA_ERROR - Standard agent for error handling in the middle tier
- WF_IN - Default inbound agent
- WF_JMS_IN - Default inbound agent for JMS Text messages
- WF_JMS_OUT - Default outbound agent for JMS Text messages
- WF_NOTIFICATION_IN - Standard inbound agent for e-mail notification responses
- WF_NOTIFICATION_OUT - Standard outbound agent for e-mail notifications
- WF_OUT - Default outbound agent
- WF_WS_JMS_IN - Default inbound agent for Web service messages
- WF_WS_JMS_OUT - Default outbound agent for Web service messages

These agents use standard queues that are automatically defined when you install Oracle Workflow. See: Setting Up Queues, *Oracle Workflow Administrator's Guide*.

You can enable or disable the WF_IN, WF_JMS_IN, WF_WS_JMS_IN, WF_OUT, WF_JMS_OUT, and WF_WS_JMS_OUT agents, but you must not make any other changes to their definitions. You must not make any changes to the definitions of the other agents.

Oracle Workflow automatically runs PL/SQL agent listeners for the standard WF_DEFERRED, WF_ERROR, and WF_NOTIFICATION_IN agents in order to perform deferred subscription processing, error handling for the Business Event System, and inbound e-mail processing for notification mailers, respectively. Oracle Workflow also automatically runs Java agent listeners for the standard WF_JAVA_DEFERRED and WF_JAVA_ERROR agents in order to perform deferred subscription processing and error handling in the middle tier. Additionally, Oracle Workflow provides a Java agent listener named Web Services IN Agent that you can optionally start for the WF_WS_JMS_IN agent. If you want to use the WF_IN and WF_JMS_IN agents for event message propagation, schedule listeners for those agents as well.

Likewise, if you want to use the WF_OUT and WF_JMS_OUT agents for event message propagation, schedule propagation for those agents. You do not need to schedule propagation for the WF_CONTROL, WF_NOTIFICATION_OUT, or WF_WS_JMS_OUT agents, however. The middle tier processes that use WF_CONTROL dequeue messages directly from its queue, and notification mailers send messages placed on the WF_NOTIFICATION_OUT queue. For WF_WS_JMS_OUT, you can optionally start a Web services outbound component named Web Services OUT Agent, provided by Oracle Workflow. For more information, please refer to the Oracle Applications Manager online help, Oracle Enterprise Manager online help, Oracle Enterprise Manager Support, *Oracle Application Developer's Guide - Advanced Queuing* or *Oracle Streams Advanced Queuing User's Guide and Reference*, Distributed Management, *Oracle Enterprise Manager Administrator's Guide*, and Setting Up the Business Event System, *Oracle Workflow Administrator's Guide*.

You can run a diagnostic test through Oracle Diagnostics to verify that the WF_DEFERRED queue and the WF_ERROR each have only one subscriber rule defined. No custom subscribers should be added to these queues. See: Oracle Workflow Diagnostic Tests, *Oracle Workflow Administrator's Guide*.

Note: Oracle Workflow also includes three additional agents named WF_REPLAY_IN, WF_REPLAY_OUT, and WF_SMTP_O_1_QUEUE, which are not currently used.

The following table lists the default properties for the standard WF_CONTROL agent. See: Cleaning Up the Workflow Control Queue, *Oracle Workflow Administrator's Guide*.

WF_CONTROL Agent Properties

Agent Property	Value
Name	WF_CONTROL
Display Name	Workflow Control Out Queue
Description	Workflow JMS Text Message Queue used to signal messages to middle tier processes
Protocol	SQLNET
Address	*<workflow schema>*.WF_CONTROL@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_CONTROL
Direction	Out
Status	Enabled

The following table lists the default properties for the standard WF_DEFERRED agent.

WF_DEFERRED Agent Properties

Agent Property	Value
Name	WF_DEFERRED
Display Name	WF_DEFERRED
Description	WF_DEFERRED
Protocol	SQLNET
Address	*<workflow schema>*.WF_DEFERRED@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_QH
Queue Name	*<workflow schema>*.WF_DEFERRED
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_ERROR agent.

WF_ERROR Agent Properties

Agent Property	Value
Name	WF_ERROR
Display Name	WF_ERROR
Description	WF_ERROR
Protocol	SQLNET
Address	*<workflow schema>*.WF_ERROR@*<local database>*
System	*<local system>*
Queue Handler	WF_ERROR_QH
Queue Name	*<workflow schema>*.WF_ERROR
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_JAVA_DEFERRED agent.

WF_JAVA_DEFERRED Agent Properties

Agent Property	Value
Name	WF_JAVA_DEFERRED
Display Name	Workflow Java Deferred In Queue
Description	Workflow Java Deferred In Queue
Protocol	SQLNET
Address	*<workflow schema>*.WF_JAVA_DEFERRED@ *<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_JAVA_DEFERRED
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_JAVA_ERROR agent.

WF_JAVA_ERROR Agent Properties

Agent Property	Value
Name	WF_JAVA_ERROR
Display Name	Workflow Java Error In Queue
Description	Workflow Java Error In Queue
Protocol	SQLNET
Address	*<workflow schema>*.WF_JAVA_ERROR@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_JAVA_ERROR
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_IN agent.

WF_IN Agent Properties

Agent Property	Value
Name	WF_IN
Display Name	WF_IN
Description	WF_IN
Protocol	SQLNET
Address	*<workflow schema>*.WF_IN@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_QH
Queue Name	*<workflow schema>*.WF_IN
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_JMS_IN agent.

WF_JMS_IN Agent Properties

Agent Property	Value
Name	WF_JMS_IN
Display Name	Workflow JMS In Queue
Description	Workflow JMS Text Message Queue
Protocol	SQLNET
Address	*<workflow schema>*.WF_JMS_IN@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_JMS_IN
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_JMS_OUT agent.

WF_JMS_OUT Agent Properties

Agent Property	Value
Name	WF_JMS_OUT
Display Name	Workflow JMS Out Queue
Description	Workflow JMS Text Message Queue
Protocol	SQLNET
Address	*<workflow schema>*.WF_JMS_OUT@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_JMS_OUT
Direction	Out
Status	Enabled

The following table lists the default properties for the standard WF_NOTIFICATION_IN agent. See: Implementing Notification Mailers, *Oracle Workflow Administrator's Guide*.

WF_NOTIFICATION_IN Agent Properties

Agent Property	Value
Name	WF_NOTIFICATION_IN
Display Name	Workflow Notification In Queue
Description	Workflow inbound notification response queue
Protocol	SQLNET
Address	*<workflow schema>*.WF_NOTIFICATION_IN@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_NOTIFICATION_IN
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_NOTIFICATION_OUT agent. See: Implementing Notification Mailers, *Oracle Workflow Administrator's Guide*.

WF_NOTIFICATION_OUT Agent Properties

Agent Property	Value
Name	WF_NOTIFICATION_OUT
Display Name	Workflow Notification Out Queue
Description	Workflow notification outbound queue
Protocol	SQLNET
Address	*<workflow schema>*.WF_NOTIFICATION_OUT@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_NOTIFICATION_OUT
Direction	Out
Status	Enabled

The following table lists the default properties for the standard WF_OUT agent.

WF_OUT Agent Properties

Agent Property	Value
Name	WF_OUT
Display Name	WF_OUT
Description	WF_OUT
Protocol	SQLNET
Address	*<workflow schema>*.WF_OUT@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_QH
Queue Name	*<workflow schema>*.WF_OUT
Direction	Out
Status	Enabled

The following table lists the default properties for the standard WF_WS_JMS_IN agent.

WF_WS_JMS_IN Agent Properties

Agent Property	Value
Name	WF_WS_JMS_IN
Display Name	WebServices JMS In Queue
Description	WebServices JMS Text Message Queue for Inbound
Protocol	SOAP
Address	*<workflow schema>*.WF_WS_JMS_IN@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_WS_JMS_IN
Direction	In
Status	Enabled

The following table lists the default properties for the standard WF_WS_JMS_OUT agent.

WF_WS_JMS_OUT Agent Properties

Agent Property	Value
Name	WF_WS_JMS_OUT
Display Name	WebServices JMS Out Queue
Description	WebServices JMS Text Message Queue for Outbound
Protocol	SOAP
Address	*<workflow schema>*.WF_WS_JMS_OUT@*<local database>*
System	*<local system>*
Queue Handler	WF_EVENT_OJMSTEXT_QH
Queue Name	*<workflow schema>*.WF_WS_JMS_OUT
Direction	Out
Status	Enabled

Agents on External Systems

Systems that will communicate events with each other must store each other's inbound agent definitions in order to address event messages to each other.

For communication between two systems that both have Oracle Workflow installed, Oracle Workflow provides an external system registration procedure that you

can use to automatically copy the inbound agent definitions for the other system into the Event Manager of your local system. See: Registering External Systems, page 8-93.

If your local Workflow-enabled system will communicate with a non-Workflow system, the non-Workflow system must provide its own external propagation agents to handle Business Event System event messages.

- An inbound agent on a non-Workflow system must be able to dequeue event messages from a Business Event System outbound queue and process the contents of those messages. The inbound agent must have an address at which systems can communicate with the agent.
- An outbound agent on a non-Workflow system must be able to enqueue event messages in the appropriate format on a Business Event System inbound queue.

You must manually define the inbound agents for an external non-Workflow system in the Event Manager of your local system. You can optionally define the external system's outbound agents as well.

1. Before defining agents for a non-Workflow system, you must define the system itself using the Create System page. See: To Create or Update a System, page 8-97.
2. You can then use the Create Agent page to define an agent for the non-Workflow system, following the same steps as for any other agent. See: To Create or Update an Agent, page 8-89.
   - You must associate the agent with the non-Workflow system to which it belongs.
   - You must specify the protocol by which you will communicate with the agent.
   - For an inbound agent, you must also specify the address at which you will communicate with the agent.
   - You can leave the queue name and queue handler blank if the agent is not implemented as an Oracle Advanced Queueing queue.

Related Topics

To View and Maintain Agents, page 8-88

To Create or Update an Agent, page 8-89

To Create or Update an Agent Group, page 8-91

Defining Agents

To View and Maintain Agents:

1. Use a Web browser to navigate to the Event Manager, using a responsibility and navigation path specified by your system administrator. Then choose Agents in the horizontal navigation. See: Oracle Workflow Developer Navigation Paths, page A-1.
2. Search for the agents you want to display. The main search option is:
   - Name - Enter the internal name of the agent you want to display. You can enter a partial value to search for agents whose internal names contain that value.

   You can also enter the following additional search criteria.
   - Protocol - Select the protocol of the agent you want to display.

- Address - Enter the address of the agent you want to display.
- System - Select the system of the agent you want to display.
- Direction - Select In or Out as the direction of the agents you want to display.
- Type - Select Agent or Group as the type of the agents you want to display.
- Status - Select Enabled or Disabled as the status of the agents you want to display.

3. If you have workflow administrator privileges, you can use the administration icons and buttons to perform administrative operations.
   - To update an agent, choose the update icon for that agent. See: To Create or Update an Agent, page 8-89 and To Create or Update an Agent Group, page 8-91.
   - To delete an agent, select the agent and select the Delete button.

     Note: You can only delete agents that do not have any subscriptions referencing them and that do not belong to any agent groups.

   - To create a new individual agent, select the Create Agent button. See: To Create or Update an Agent, page 8-89.
   - To create a new agent group, select the Create Agent Group button. See: To Create or Update an Agent Group, page 8-91.

     Note: Whenever you make changes to your agents that affect the physical implementation required for message propagation, you should recheck your propagation setup. For more information, please refer to the Oracle Applications Manager online help, Oracle Enterprise Manager online help, Oracle Enterprise Manager Support, *Oracle Application Developer's Guide - Advanced Queuing* or *Oracle Streams Advanced Queuing User's Guide and Reference,* Distributed Management, *Oracle Enterprise Manager Administrator's Guide,* and Setting Up the Business Event System, *Oracle Workflow Administrator's Guide*.

To Create or Update an Agent:

1. Navigate to the Create Agent page or to the Update Agent page. The Create Agent page and the Update Agent page are identical, except that the fields in the Update Agent page are populated with previously defined information for the selected agent.

   Note: You should not use the Create Agent page to create agent definitions for inbound agents on external systems that have Oracle Workflow installed. Instead, you should use the External System Registration page to automatically register the system identifier information for that external system, including inbound agent definitions. See: Registering External Systems, page 8-93.

2. Enter the internal name of the agent in the Name field. The agent's internal name must be unique within the agent's system.

   Important: The internal name must be all-uppercase and should not include any single or double quotation marks (`'` or `"`) or spaces.

3. Enter a display name for the agent.

4. Enter an optional description for the agent.
5. Select the message communication protocol that the agent supports.
6. If the agent supports inbound communication to its system, enter the address for the agent. The format of the address depends on the protocol you select.

   For agents that use the SQLNET protocol, the address must be in the following format to enable AQ propagation:

   ```
   <schema>.<queue>@<database link>
   ```

   <schema> represents the schema that owns the queue, *<queue>* represents the queue name, and *<database link>* represents the database link to the instance where the queue is located.

   > **Note:** You must enter the database link name exactly as the name was specified when the database link was created. See: Creating Database Links, *Oracle Workflow Administrator's Guide*.

7. Select the system in which the agent is defined.
8. Enter the queue handler for the agent. A queue handler is a PL/SQL or Java package that translates between the Workflow event message format (`WF_EVENT_T` datatype or `BusinessEvent` Java object) and the message format required by the queue associated with the agent. Define only one queue handler for an agent, either PL/SQL or Java. See: Standard APIs for a Queue Handler, page 6-21.
   - To assign the agent a PL/SQL queue handler, enter the PL/SQL package name in the Queue Handler field in all uppercase. You can run both PL/SQL and Java agent listeners on an agent with a PL/SQL queue handler.
   - To assign the agent a Java queue handler, enter the Java package name in the Java Queue Handler field. You can run only Java agent listeners on an agent with a Java queue handler.

     > **Note:** For an agent that is not implemented as a queue, such as an agent on a non-Oracle system, you can leave the Queue Handler and Java Queue Handler fields blank.

9. Enter the name of the queue that the local system uses to interact with the agent. Since only the local system refers to this queue name, the queue name should be within the scope of this system, without requiring a database link. Use the following format to specify the queue name:

   ```
   <schema>.<queue>
   ```

   <schema> represents the schema that owns the queue, and *<queue>* represents the queue name.

   > **Important:** You must enter the queue name in all uppercase.

   > **Note:** For an agent that is not implemented as a queue, such as an agent on a non-Oracle system, you can leave the Queue field blank.

10. In the Direction field, select In for an agent that supports inbound communication to its system, or select Out for an agent that supports outbound communication from its system.
11. Select Enabled or Disabled as the agent status. If you disable an agent, its definition remains in the Event Manager for reference, but you cannot use the agent in active subscriptions.

To Create or Update an Agent Group:

1. Navigate to the Create Agent Group page or to the Update Agent Group page. The Create Agent Group page and the Update Agent Group page are identical, except that the fields in the Update Agent Group page are populated with previously defined information for the selected agent group.
2. Enter the internal name of the agent group in the Name field. The agent group's internal name must be unique within the agent group's system.

   Important: The internal name must be all-uppercase and should not include any single or double quotation marks (`'` or `"`) or spaces.

3. Enter a display name for the agent group.
4. Enter an optional description for the agent group.
5. Select the system on which the agent group is defined.

   Note: Although an agent group is defined on a particular system, you can include agents from other systems as members within the group.

   Since agent groups are used only for inbound communication, the direction for an agent group is automatically set to In.
6. Select Enabled or Disabled as the agent group status. If you disable an agent group, its definition remains in the Event Manager for reference, but you cannot use the agent group in active subscriptions.
7. Select the Apply button to save the agent group definition.
8. To add a member agent to the group, select the Add Agents to Group button.

   Note: An agent group can contain only individual agents as its members. It cannot contain another group.

9. In the Add Agents to Group page, search for the agents you want to add. The main search option is:
   - Name - Enter the internal name of the agent you want to add. You can enter a partial value to search for agents whose internal names contain that value.

   You can also enter the following additional search criteria.
   - Protocol - Select the protocol of the agent you want to add.
   - Address - Enter the address of the agent you want to add.
   - System - Select the system of the agent you want to add.
   - Status - Select Enabled or Disabled as the status of the agents you want to add.

10. Select the agent or agents that you want to add to your agent group, and select the Add Agents to Group button.
11. You can optionally enter new search criteria to search for other agents to add to the agent group.
12. After you finish adding agents to the agent group, select the Apply button to save the agent group members.
13. To delete a member agent from the group, select the agent in the Create Agent Group or Update Agent Group page and select the Delete button.

    Note: Deleting a member agent from an agent group does not delete the agent definition for the individual agent. The individual agent remains in the Event Manager.

Systems

A system is a logically isolated software environment such as a host machine or database instance. You should define each system to or from which you will communicate events in the Event Manager.

When you define a system, you can specify whether it is associated with a master system from which you want it to receive Event Manager object definition updates.

Each system can expose one or more addressable points of communication, called agents. After you define your systems, you should define the agents within those systems that you will use to communicate business events. See: Agents, page 8-76.

Local System

When you install Oracle Workflow in a database, that database is automatically defined as a system in the Event Manager and set as the local system in the Workflow Configuration page. The following table lists the default properties of the local system definition.

Local System Properties

System Property	Value
Name	*<database global name>*
Display Name	*<database global name>*
Description	Local System Created by Oracle Workflow Configuration Assistant
Master	(blank)

You can update the local system definition if necessary.

Oracle Workflow sets the status of the local system to Enabled by default. After you finish setting up the Business Event System, you can use the Workflow Configuration page to set the system status that you want for event processing. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

Registering External Systems

Before you can send business events from one system to another, you must register the destination system with the source system as a potential recipient of event messages. Registering a system means defining the destination system as well as its inbound agents in the Event Manager of the source system, so that event messages from the source system can be addressed to the destination agents. Registering a system is also known as signing up a system.

Usually, both systems should be registered with each other, so that each system can both send messages to and receive messages from the other system.

Oracle Workflow provides Web pages to help automate external system registration between two systems that both have Oracle Workflow installed. For communication between a Workflow-enabled system and a non-Workflow system, you must perform manual steps to store the required destination system and agent information in the source system.

To register a destination system for receiving event messages from a source system, perform the following steps:

1. Retrieve the local system and inbound agent definitions, which together make up the system identifier information, from the destination system.

   Note: The system identifier information includes only the definitions for the system itself and its individual inbound agents. The system identifier does not include agent group definitions.

   - If Oracle Workflow is installed on the destination system, you can use the Local System Identifier page on the destination system to generate and save an XML document containing the system identifier information. See: To Retrieve Local System Identifier Information, page 8-97.

     Note: If you do not have access to the Oracle Workflow installation on the destination system, ask the workflow administrator for that system to perform this step.

   - If Oracle Workflow is not installed on the destination system, you will need to manually gather the information needed to create definitions for the system and its inbound agents.

2. Register the destination system identifier information in the Event Manager in the source system.

   - If Oracle Workflow is installed on both the source system and the destination system, you can use the External System Registration Web page on the source system to register the information by raising the System Signup event with the system identifier XML document from the destination system as the event data. When the System Signup event is raised on the source system, Oracle Workflow executes a predefined subscription that loads the system identifier information to the Event Manager in that system. See: To Register an External System, page 8-98.

     Note: If you do not have access to the Oracle Workflow installation on the source system, ask the workflow administrator for that system to perform this step.

- If Oracle Workflow is installed on the source system but not on the destination system, you must manually create system and agent definitions for the destination system in the Event Manager of the source system. See: To Create or Update a System, page 8-97 and To Create or Update an Agent, page 8-89.
- If Oracle Workflow is not installed on the source system but is installed on the destination system, you must store the system and inbound agent information for the destination system in the source system according to that non-Workflow source system's requirements. You can optionally make use of the system identifier XML document generated by the Local System Identifier page on the destination system, but you must manually perform whatever steps are necessary to store that information in the non-Workflow source system.

Synchronizing Systems

Synchronizing systems means replicating all the Event Manager objects that are defined on the source system to the target system. You can use the Synchronize Event Systems event to synchronize two Workflow-enabled systems with each other.

To synchronize systems, perform the following steps:

1. Register the source and target systems with each other. See: Registering External Systems, page 8-93.
2. On the source system, copy the predefined subscription to the Seed Event Group with the Local source type and modify the copy as follows.
   - Specify the inbound agent on the target system that you want to receive the event message, or specify a workflow process that sends the event message to the target system.

     Note: If you want to send the event message to more than one target system, you can specify an agent group to receive the event message.

   - Enable the subscription.

     Note: If you do not have access to the Oracle Workflow installation on the source system, ask the workflow administrator for that system to perform this step.

3. On the target system, enable the predefined subscription to the Seed Event Group with the External source type.

   Note: If you do not have access to the Oracle Workflow installation on the target system, ask the workflow administrator for that system to perform this step.

4. On the source system, raise the Synchronize Event Systems event (`oracle.apps.wf.event.all.sync`) using the Test Business Event page for the event. Enter a unique event key, but do not enter any event data. See: To Raise a Test Event, page 8-57.

   Note: If you do not have access to the Oracle Workflow installation on the source system, ask the workflow administrator for that system to perform this step.

When the Synchronize Event Systems event is raised on the source system, it triggers the subscription to the Seed Event Group with the Local source type. The Event Manager generates the event message, which contains the definitions of all the Event Manager objects on the local system, including events, event groups, systems, agents, agent groups, and subscriptions. Then the event message is sent to the specified inbound agent on the target system, or to the specified workflow process that sends the event message to the target system.

When the Synchronize Event Systems event is received on the target system, it triggers the subscription to the Seed Event Group with the External source type. Oracle Workflow loads the object definitions from the event message into the Event Manager on the target system, creating new definitions or updating existing definitions as necessary.

Automatic Replication

After you enable the predefined subscriptions in steps 2 and 3, these subscriptions will also replicate any subsequent changes you make to Event Manager object definitions on the source system. Whenever you create, update, or delete events, event group members, systems, agents, agent group members, or subscriptions, Oracle Workflow raises the corresponding predefined events. These events trigger the Local subscription to the Seed Event Group on the source system, which sends the object definition data to the target system. The External subscription to the Seed Event Group on the target system receives the data and adds, updates, or deletes the object definition in the Event Manager there.

If you do not want to continue automatically replicating changes on the source system to the target system, you can either disable the subscriptions after you finish synchronizing the systems, or disable the predefined events corresponding to those changes.

Master/Copy Systems

If you choose, you can treat one system as a master system that replicates its own Event Manager object definitions to its associated copy systems, but does not accept any object definition changes from those systems. To set up master/copy replication, perform the steps to synchronize the target copy systems with the source master system, as usual. Then, to prevent object definitions from being sent from the copy systems, ensure that the Local subscription to the Seed Event Group on the copy systems is disabled. To prevent object definitions from being received into the master system, ensure that the External subscription to the Seed Event Group on the master system is disabled as well.

Related Topics

To View and Maintain Systems, page 8-96

To Create or Update a System, page 8-97

To Retrieve Local System Identifier Information, page 8-97

To Register an External System, page 8-98

Predefined Workflow Events, page 9-1

Synchronize Event Systems Event, page 9-9

Seed Event Group, page 9-9

To View and Maintain Event Subscriptions, page 8-70

Defining Systems

To View and Maintain Systems:

1. Use a Web browser to navigate to the Event Manager, using a responsibility and navigation path specified by your system administrator. Then choose Systems in the horizontal navigation, and choose System Details in the side navigation. See: Oracle Workflow Developer Navigation Paths, page A-1.

2. Search for the systems you want to display. The main search option is:

   - Name - Enter the internal name of the system you want to display. You can enter a partial value to search for systems whose internal names contain that value.

   You can also enter the following additional search criteria:

   - Display Name - Enter the display name of the system you want to display.
   - Master - Select the master system for the system you want to display.

   If you display the local system, its internal name is marked with an asterisk (*).

3. To view the agents on a system, select the agents icon for that system.

   If you have workflow administrator privileges, you can update an agent by selecting the update icon for that agent. See: To Create or Update an Agent, page 8-89.

4. To view the subscriptions for a system, select the subscription icon for that system.

   > **Note:** For systems that do not have any subscriptions, a blank subscription icon appears in the Subscription column. For systems that do have subscriptions, a full subscription icon appears.

   - If you have workflow administrator privileges, you can define a new subscription for the system by selecting the Create Subscription button. The Create Event Subscription page appears with the system name automatically entered as the subscriber in the System field. See: To Create or Update an Event Subscription, page 8-71.
   - If you have workflow administrator privileges, you can update an existing subscription by selecting the update icon for that subscription. See: To Create or Update an Event Subscription, page 8-71.

     > **Note:** For seeded subscriptions with a customization level of Limit, you can only update the subscription status. For seeded subscriptions with a customization level of Core, you cannot update any properties; you can only view the subscription definition.

5. If you have workflow administrator privileges, you can use the administration icons and buttons in the System Details page to perform administrative operations.

   - To update a system, choose the update icon for that system. See: To Create or Update a System, page 8-97.
   - To delete a system, select the system and select the Delete button.

     > **Note:** You can only delete systems that do not have any agents defined on them or any subscriptions referencing them. Also, you cannot delete the local system.

- To create a new system definition, select the Create System button. See: To Create or Update a System, page 8-97.

  Note: You should use the Create System page only to manually create system definitions for external systems that are not Workflow-enabled. For a Workflow-enabled system, you should use the External System Registration page instead to automatically register the system identifier information. See: Registering External Systems, page 8-93.

To Create or Update a System:

1. Navigate to the Create System page or to the Update System page. The Create System page and the Update System page are identical, except that the fields in the Update System page are populated with previously defined information for the selected system.

   Note: You should use the Create System page only to manually create system definitions for external systems that are not Workflow-enabled. For a Workflow-enabled system, you should use the External System Registration page instead to automatically register the system identifier information. See: Registering External Systems, page 8-93.

   You can use the Update System page to update both systems that are Workflow-enabled and those that are not.

2. Enter the internal name of the system in the Name field.

   Important: The internal name must be all-uppercase and should not include any single or double quotation marks (' or ") or spaces.

3. Enter a display name for the system.
4. Enter an optional description for the system.
5. Optionally enter a master system from which you want this system to receive Event Manager object definition updates.

To Retrieve Local System Identifier Information:

1. Use a Web browser to navigate to the Event Manager on the system you want to register as a destination system, using a responsibility and navigation path specified by your system administrator. Then choose Systems in the horizontal navigation, and choose Local System Identifier in the side navigation. See: Oracle Workflow Developer Navigation Paths, page A-1.

   Note: If you do not have access to the Oracle Workflow installation on the destination system, ask the workflow administrator for that system to perform this step.

2. Select the Generate button. Oracle Workflow generates the local system identifier XML document, which contains the definitions of the local system and its inbound agents, and displays that document in the XML Content field.

3. Select the Save button to save the XML document to your file system as an XML file. You can then enter it as the event data when you register this system with a source system. See: To Register an External System, page 8-98.

To Register an External System:

1. Use a Web browser to navigate to the Event Manager on the source system where you want to register a destination system, using a responsibility and navigation path specified by your system administrator. Then choose Systems in the horizontal navigation, and choose External System Registration in the side navigation. See: Oracle Workflow Developer Navigation Paths, page A-1.

   Note: If you do not have access to the Oracle Workflow installation on the source system, ask the workflow administrator for that system to perform this step.

2. In the Event Identifier region, specify the identifying information for the System Signup event.
   - Event Name - Oracle Workflow automatically displays the internal name of the System Signup event, `oracle.apps.wf.event.system.signup`.
   - Event Key - Enter an event key that uniquely identifies this instance of the event.
   - Send Date - Optionally specify the date when the event message is available for dequeuing. If you set the send date to a future date when you raise the event, the event message is placed on the WF_DEFERRED queue, and subscription processing does not begin until the specified date.
3. In the Event Parameter region, optionally enter any additional parameter name and value pairs to be stored in the parameter list within the event message. You can enter up to 100 parameters.
4. In the Event Data region, enter the system identifier XML document for the destination system you want to register. See: To Retrieve Local System Identifier Information, page 8-97.
   - To enter the system identifier information manually, select the Write XML option and paste the XML document into the XML Content field. You can enter up to 4000 characters.
   - To upload the system identifier XML file from your file system, select the Upload XML option and select the Upload XML File button. In the Upload XML File page, enter the full path and file name of the XML file containing the system identifier information, and select the Upload File button.

     You can also choose the View link to view the XML file if you have already specified one, or choose the Clear button to remove the previously specified file and enter a different file.

     Note: You must provide the system identifier information in order to register a system. You cannot leave the event data XML content blank.

5. Choose the Submit button to raise the System Signup event to the Event Manager. When the System Signup event is raised, Oracle Workflow executes a

predefined subscription that loads the system identifier information from the event data to the Event Manager. See: System Signup Event, page 9-13.

Workflow Agent Ping/Acknowledge

You can test your Business Event System setup using the Workflow Agent Ping/Acknowledge workflow. This workflow sends a ping event message to each inbound agent on the local system or on external systems, and waits to receive an acknowledgement event message from each of the agents. If the workflow completes successfully, then the basic Business Event System setup for communication with these agents is complete.

How Workflow Agent Ping/Acknowledge Works

Use the Launch Processes Web page to launch the Workflow Agent Ping/Acknowledge workflow. This workflow consists of two processes, the Master Ping process and the Detail Ping process. To ping all inbound agents, select the Master Ping process, and enter a unique item key. See: Testing Workflow Definitions, page 7-1.

> **Note:** The item key for a process instance can only contain single-byte characters. It cannot contain a multibyte value.

When you launch the Master Ping process, the Workflow Engine identifies all the inbound agents that you have defined on the local system or on external systems and launches a Detail Ping process for each agent. The master process then waits for each detail process to complete.

The Detail Ping process begins by sending a Ping Agent event to the inbound agent identified by the master process. The detail process places a Ping Agent event message on a queue associated with an outbound agent on the local system. The event message is addressed to the inbound agent and contains a correlation ID that identifies the detail process to which it belongs. AQ propagation transmits the event message from the outbound queue to the queue associated with the specified inbound agent.

On the receiving system, the listener for the inbound agent dequeues the Ping Agent message the next time it runs. When the event message is dequeued, the Event Manager searches for and executes any active subscriptions to the Ping Agent event or the Any event on that system that have a source type of External.

When the predefined External subscription to the Ping Agent event is executed, its rule function places an Acknowledge Ping event message on a queue associated with an outbound agent on that system. The event message is addressed to an inbound agent on the originating system and includes the correlation ID from the Ping Agent event message. AQ propagation transmits the Acknowledge Ping event message from the outbound queue to the queue associated with the specified inbound agent.

On the originating system, the listener for the inbound agent dequeues the Acknowledge Ping message the next time it runs. When the event message is dequeued, the Event Manager searches for and executes any active subscriptions to the Acknowledge Ping event or the Any event on that system that have a source type of External.

When the predefined External subscription to the Acknowledge Ping event is executed, its rule function, which is the default rule function, sends the event message to the Detail Ping process. The Workflow Engine uses the correlation ID to match the

message with the running detail process to which it belongs. After receiving the event message, the Detail Ping process completes.

Finally, after all the detail processes are complete, the master process also completes.

You can use the Workflow Monitor to check the progress of the Workflow Agent Ping/Acknowledge workflow. You can also use Oracle Workflow Manager to confirm the processing of the Ping Agent and Acknowledge Ping event messages. See: Workflow Monitor, *Oracle Workflow Administrator's Guide* and the Oracle Workflow Manager online help.

The amount of time needed to complete the Workflow Agent Ping/Acknowledge workflow depends on how often the listeners run to dequeue messages from the inbound agents. See: Scheduling Listeners for Local Inbound Agents, *Oracle Workflow Administrator's Guide*.

Related Topics

Ping Agent Events, page 9-12

The Workflow Agent Ping/Acknowledge Item Type

The Workflow Agent Ping/Acknowledge process is associated with an item type called Workflow Agent Ping/Acknowledge. Currently there are two workflow processes associated with Workflow Agent Ping/Acknowledge: Master Ping Process and Detail Ping Process.

To view the details of the Workflow Agent Ping/Acknowledge item type in the Workflow Builder, choose Open from the File menu. Then connect to the database and select the Workflow Agent Ping/Acknowledge item type, or connect to a file called `wfping.wft` in the *<ORACLE_HOME>*`\wf\Data\`*<language>* directory on your file system.

If you examine the property page of Workflow Agent Ping/Acknowledge, you see that it has a persistence type of Temporary and persistence number of days of 0. This means that the runtime data associated with any work items for this item type are eligible for purging as soon as they complete.

The Workflow Agent Ping/Acknowledge item type also has several attributes associated with it. These attributes reference information in the Workflow application tables. The attributes are used and maintained by function activities as well as event activities throughout the process. The following table lists the Workflow Agent Ping/Acknowledge item type attributes.

Workflow Agent Ping/Acknowledge Item Type Attributes

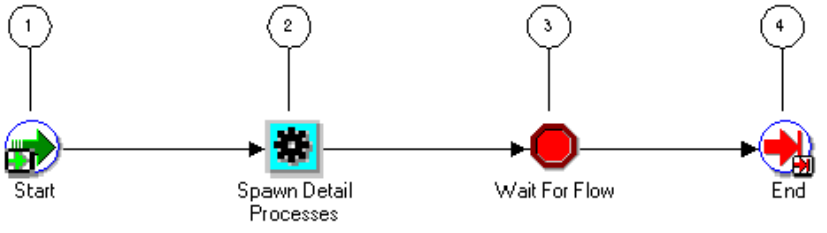
Display Name	Description	Type	Length/Format/ Lookup Type
To Agent	The inbound agent that receives the event message, in the format *<agent>@<system>*	Text	
Event Name	The internal name of the event	Text	
Out Agent	The outbound agent that sends the event message, in the format *<agent>@<system>*	Text	
Event Key	The event key that uniquely identifies the specific instance of the event	Text	
Event Message	The event message	Event	

Summary of the Master Ping Process

To view the properties of the Master Ping process, select the process in the navigator tree, and then choose Properties from the Edit menu. This process activity is runnable, indicating that it can be initiated as a top level process to run.

When you display the Process window for the Master Ping process, you see that the process consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Master Ping Process Diagram



The Workflow Agent Ping/Acknowledge workflow begins when you launch the Master Ping Process using the Launch Processes Web page. You can optionally provide a to agent, event name, out agent, event key, and event message. See: Testing Workflow Definitions, page 7-1.

The workflow begins at node 1 with the Start activity. At node 2, the master process spawns a detail process for each inbound agent that you have defined on the local system or on external systems. The detail process pings the agent by sending it a Ping

Agent event and waits to receive an acknowledgement in the form of an Acknowledge Ping event.

Node 3 is a Wait for Flow activity that waits for all the detail processes to complete. When all the detail processes have completed, the master process ends.

Master Ping Process Activities

Following is a description of each activity in the process, listed by the activity's display name.

Start (Node 1)

This Standard function activity marks the start of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

Spawn Detail Processes (Node 2)

This function activity identifies all the inbound agents that you have defined on the local system or external systems, and spawns a Detail Ping process for each agent. The function sets the Ping Agent event (`oracle.apps.wf.event.test.ping`) as the event to be sent to the Detail Ping processes.

Function
WF_EVENT_PING_PKG.LAUNCH_PROCESSES

Result Type
None

Prerequisite Activities
None

Item Attributes Set by Function
Event Name, To Agent

Wait for Flow (Node 3)

This Standard function activity pauses the flow until the corresponding detail processes complete a specified activity.

Function
WF_STANDARD.WAITFORFLOW

Result Type
None

Prerequisite Activities
Spawn Detail Processes

End (Node 4)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

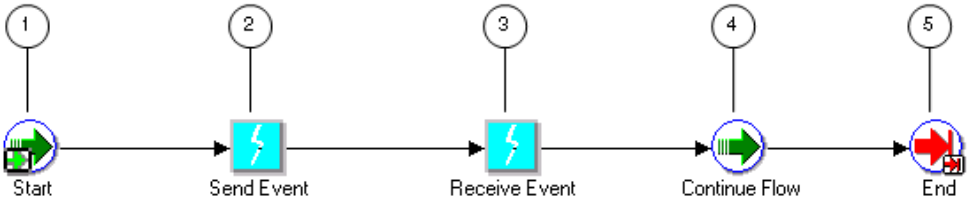
Prerequisite Activities
Wait for Flow

Summary of the Detail Ping Process

To view the properties of the Detail Ping process, select its process activity in the navigator tree, and then choose Properties from the Edit menu. This process activity is runnable, indicating that it can be initiated as a top level process to run.

When you display the Process window for the Detail Ping process, you see that the process consists of five unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Detail Ping Process Diagram



The Detail Ping process begins when it is launched by the Master Ping process. See: Summary of the Master Ping Process, page 8-101.

The workflow begins at node 1 with the Start activity. At node 2, the process sends a Ping Agent event to the selected inbound agent. At node 3, the process waits to receive an Acknowledge Ping event back from the agent. When the acknowledgement is received, the master process can continue. The detail process ends at this point.

Detail Ping Process Activities

Following is a description of each activity in the process, listed by the activity's display name.

Start (Node 1)

This Standard function activity marks the start of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

Send Event (Node 2)

This event activity sends the Ping Agent event (`oracle.apps.wf.event.test.ping`) from an outbound agent on the local system to the inbound agent identified by the master process. The event message includes a correlation ID that identifies the detail process to which it belongs.

Event Action
Send

Prerequisite Activities
None

Item Attributes Retrieved by Activity
Event Message, Event Name, Event Key, To Agent

Receive Event (Node 3)

This event activity receives the Acknowledge Ping event (`oracle.apps.wf.event.test.ack`) that is returned to the originating system from the system that received the Ping Agent event. The Acknowledge Ping event message contains the correlation ID, which the Workflow Engine uses to match the event message with the detail process to which it belongs.

Event Action
Receive

Event Filter
`oracle.apps.wf.event.test.ack`

Prerequisite Activities
Send Event

Item Attributes Set by Activity
Event Name, Event Key, Event Message

Continue Flow (Node 4)

This Standard function activity marks the position in the detail process where, upon completion, the corresponding halted master process will continue.

Function
WF_STANDARD.CONTINUEFLOW

Result Type
None

Prerequisite Activities
Receive Event

End (Node 5)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
Continue Flow

9

Predefined Workflow Events

This chapter tells you how to use Oracle Workflow's predefined events.

This chapter covers the following topics:

- Predefined Workflow Events
- Workflow Send Protocol

Predefined Workflow Events

Oracle Workflow provides several predefined events for significant occurrences within the Business Event System and other Oracle Workflow components. You can define subscriptions to these events for replication, validation, or other purposes. You can enable or disable many of the events if necessary.

Some predefined events are referenced by default subscriptions that are created automatically when you install Oracle Workflow. The subscriber for all the default subscriptions is the local system. You can enable, disable, or copy many of these subscriptions to perform the event processing that you want.

> **Important:** You must not change or disable the definition of the Unexpected event or of the predefined Error subscription to that event. If you do, the Event Manager will not be able to perform default error handling for events and subscription processing.

> **Note:** Predefined events and default subscriptions are also provided with some Oracle Applications products. For more information on Oracle Applications-specific workflow events, consult the documentation or help for that specific Oracle Application product and the Oracle E-Business Suite Electronic Technical Reference Manual, available from Oracle*MetaLink*.

Related Topics

Event Definition Events, page 9-2

Event Group Definition Events, page 9-3

System Definition Events, page 9-4

Agent Definition Events, page 9-5

Agent Group Definition Events, page 9-7

Event Subscription Definition Events, page 9-8

Synchronize Event Systems Event, page 9-9

Seed Event Group, page 9-9

Ping Agent Events, page 9-12

System Signup Event, page 9-13

Any Event, page 9-14

Unexpected Event, page 9-17

User Entry Has Changed Event, page 9-19

Notification Events, page 9-21

Notification Mailer Events, page 9-30

Business Event System Control Events, page 9-32

Generic Service Component Framework Control Events, page 9-34

Workflow Engine Events, page 9-37

Directory Service Events, page 9-38

Workflow Send Protocol, page 9-45

Event Definition Events

Event Created

Oracle Workflow raises this event whenever a new individual event or event group definition is created.

Internal Name
oracle.apps.wf.event.event.create

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Event Updated

Oracle Workflow raises this event whenever an individual event or event group definition is updated.

Internal Name
oracle.apps.wf.event.event.update

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Event Deleted

Oracle Workflow raises this event whenever an individual event or event group definition is deleted.

Internal Name
oracle.apps.wf.event.event.delete

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Event Group Definition Events

Event Group Creation

Oracle Workflow raises this event whenever a new event group member definition is created.

Internal Name
oracle.apps.wf.event.group.create

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Event Group Updated

Oracle Workflow raises this event whenever an event group member definition is updated.

Internal Name
oracle.apps.wf.event.group.update

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Event Group Deleted

Oracle Workflow raises this event whenever an event group member definition is deleted.

Internal Name
oracle.apps.wf.event.group.delete

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

System Definition Events

System Created

Oracle Workflow raises this event whenever a new system definition is created.

Internal Name
oracle.apps.wf.event.system.create

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

System Updated

Oracle Workflow raises this event whenever a system definition is updated.

Internal Name
oracle.apps.wf.event.system.update

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

System Deleted

Oracle Workflow raises this event whenever a system definition is deleted.

Internal Name
oracle.apps.wf.event.system.delete

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Agent Definition Events

Agent Created

Oracle Workflow raises this event whenever a new agent definition is created.

Internal Name
oracle.apps.wf.event.agent.create

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Agent Updated

Oracle Workflow raises this event whenever an agent definition is updated.

Internal Name
oracle.apps.wf.event.agent.update

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Agent Deleted

Oracle Workflow raises this event whenever an agent definition is deleted.

Internal Name
oracle.apps.wf.event.agent.delete

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Agent Group Definition Events

Note: Agent groups are currently available only for the version of Oracle Workflow embedded in Oracle Applications.

Agent Group Member Created

Oracle Workflow raises this event whenever a new agent group member definition is created.

Internal Name
oracle.apps.wf.agent.group.create

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Agent Group Member Updated

Oracle Workflow raises this event whenever an agent group member definition is updated.

Internal Name
oracle.apps.wf.agent.group.update

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Agent Group Member Deleted

Oracle Workflow raises this event whenever an agent group member definition is deleted.

Internal Name
oracle.apps.wf.agent.group.delete

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Event Subscription Definition Events

Subscription Created

Oracle Workflow raises this event whenever a new subscription definition is created.

Internal Name
oracle.apps.wf.event.subscription.create

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Subscription Updated

Oracle Workflow raises this event whenever a subscription definition is updated.

Internal Name
oracle.apps.wf.event.subscription.update

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Subscription Deleted

Oracle Workflow raises this event whenever a subscription definition is deleted.

Internal Name
oracle.apps.wf.event.subscription.delete

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Synchronize Event Systems Event

You can raise this event to synchronize the Event Manager data on the local system with another system. The event message for the Synchronize Systems event contains the definitions of all the Event Manager objects on the local system. See: Synchronizing Systems, page 8-47.

Internal Name
oracle.apps.wf.event.all.sync

Status
Enabled

Generate Function
wf_event_functions_pkg.generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Seed Event Group

This event group contains events used for automatic replication of Business Event System objects from one system to another. The group includes all the event, event group, system, agent, and subscription definition events, as well as the Synchronize Event Systems event.

Internal Name
oracle.apps.wf.event.group.all

Status
Enabled

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Members

- oracle.apps.wf.event.event.create
- oracle.apps.wf.event.event.update
- oracle.apps.wf.event.event.delete
- oracle.apps.wf.event.group.create
- oracle.apps.wf.event.group.update
- oracle.apps.wf.event.group.delete
- oracle.apps.wf.event.system.create
- oracle.apps.wf.event.system.update
- oracle.apps.wf.event.system.delete
- oracle.apps.wf.event.agent.create
- oracle.apps.wf.event.agent.update
- oracle.apps.wf.event.agent.delete
- oracle.apps.wf.agent.group.create
- oracle.apps.wf.agent.group.update
- oracle.apps.wf.agent.group.delete
- oracle.apps.wf.event.subscription.create
- oracle.apps.wf.event.subscription.update
- oracle.apps.wf.event.subscription.delete
- oracle.apps.wf.event.all.sync

> **Note:** Agent groups are currently available only for the version of Oracle Workflow embedded in Oracle Applications.

Oracle Workflow provides two default subscriptions to the Seed Event Group. The first subscription can send the Event Manager data to an agent and to a workflow process when one of the group member events is raised locally. To use this subscription, you must create a copy of this subscription with the same properties, except you should give your new subscription a customization level of User and add the agent or workflow to which you want to send the data, and enable the subscription. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.wf.event.group.all
Phase	50
Status	Disabled
Rule Data	Message
Rule Function	wf_rule.default_rule
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The second subscription can load the Event Manager data into the local system when one of the group member events is received from an external source. To use this subscription, you must enable it. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	oracle.apps.wf.event.group.all
Phase	50
Status	Disabled
Rule Data	Key
Rule Function	wf_event_functions_pkg.receive
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Related Topics

Synchronizing Systems, page 8-47

Ping Agent Events

Ping Agent Event

The Detail Ping process in the Workflow Agent Ping/Acknowledge item type sends this event to ping inbound agents. You can use the Launch Processes Web page to launch the Master Ping Process, which in turn launches the Detail Ping process. See: Workflow Agent Ping/Acknowledge, page 8-99.

Internal Name
oracle.apps.wf.event.test.ping

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides one default subscription to the Ping Agent event. This subscription sends the Acknowledge Ping event back to the originating system when the Ping Agent event is received from an external source. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	oracle.apps.wf.event.test.ping
Phase	50
Status	Enabled
Rule Data	Key
Rule Function	wf_event_ping_pkg.acknowledge
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Acknowledge Ping

Oracle Workflow sends this event back to the originating system when a Ping Agent event is received. See: Workflow Agent Ping/Acknowledge, page 8-99.

Internal Name
oracle.apps.wf.event.test.ack

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides one default subscription to the Acknowledge Ping event. This subscription sends the Acknowledge Ping event to the Detail Ping process in the Workflow Agent Ping/Acknowledge item type when the event is received from an external source. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	oracle.apps.wf.event.test.ack
Phase	50
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.default_rule
Workflow Item Type	WFPING
Workflow Process Name	WFDTLPNG
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

System Signup Event

You can raise this event from the System Signup Web page on a source system to sign up a destination system for receiving event messages from the source system. See: To Sign Up a System, page 8-47.

Internal Name
oracle.apps.wf.event.system.signup

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides one default subscription to the System Signup event. This subscription loads the Event Manager data into the local system when the System Signup event is raised locally. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.wf.event.system.signup
Phase	50
Status	Enabled
Rule Data	Key
Rule Function	wf_event_functions_pkg.receive
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Any Event

This event is raised implicitly when any other event is raised locally or received from an external source. You can define a subscription to the Any event to implement processing that you want to execute whenever an event occurs.

Internal Name
oracle.apps.wf.event.any

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides three default subscriptions to the Any event. The first subscription can be triggered when an event is raised locally. To use this subscription, you must create a copy of this subscription with the same properties, except you should give your new subscription a customization level of User, define the action for the subscription, and enable it. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.wf.event.any
Phase	100
Status	Disabled
Rule Data	Key
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The second subscription can be triggered when an event is received from an external source. To use this subscription, you must create a copy of this subscription with the same properties, except you should give your new subscription a customization level of User, define the action for the subscription and enable it. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	oracle.apps.wf.event.any
Phase	100
Status	Disabled
Rule Data	Key
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The third subscription sends the event message to the Default Event Error process in the System: Error item type and raises an exception when an event is received from an Error source (that is, when it is dequeued from the WF_ERROR or WF_JAVA_ERROR queue). To use this subscription, you must enable it. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Error
Event Filter	oracle.apps.wf.event.any
Phase	100
Status	Disabled
Rule Data	Key
Rule Function	wf_rule.error_rule
Workflow Item Type	WFERROR
Workflow Process Name	DEFAULT_EVENT_ERROR
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Unexpected Event

Oracle Workflow executes subscriptions to this event when an event is raised locally or received from an external source, but no subscription exists for that event.

> **Important:** You must not change or disable the definition of the Unexpected event. If you do, the Event Manager will not be able to perform error handling for event and subscription processing.

Internal Name
oracle.apps.wf.event.unexpected

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides two default subscriptions to the Unexpected event. The first subscription sends the event message to the Default Event Error process in the System: Error item type when an unexpected event is received from an external source. This subscription allows your local system to handle any event messages received from external systems that you were not expecting.

The Default Event Error process notifies the system administrator, who can retry or abort subscription processing for the event. For example, the system administrator can optionally define a subscription to process the event and then retry the event.

The External subscription to the Unexpected event is enabled by default. You can disable it if necessary.

> **Important:** If you want to disable this subscription, be careful to consider all the consequences for handling unexpected event messages from external sources. If the subscription is disabled, these event messages will remain on the inbound queue where they are received and may be undetected for some time.

The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	oracle.apps.wf.event.unexpected
Phase	50
Status	Enabled
Rule Data	Key
Workflow Item Type	WFERROR
Workflow Process Name	DEFAULT_EVENT_ERROR
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The second subscription sends the event message to the Default Event Error process in the System: Error item type when an unexpected event is received from an Error source (that is, when it is dequeued from the WF_ERROR or WF_JAVA_ERROR queue). This subscription is enabled by default.

> **Important:** You must not change or disable the definition of the predefined Error subscription to the Unexpected event. If you disable this subscription, then the Event Manager will not be able to perform error handling for any events for which you have not defined custom Error subscriptions.

The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Error
Event Filter	oracle.apps.wf.event.unexpected
Phase	50
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.error_rule
Workflow Item Type	WFERROR
Workflow Process Name	DEFAULT_EVENT_ERROR
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

User Entry Has Changed Event

The Workflow LDAP APIs raise this event when changed user information is retrieved from an LDAP directory. You can run these APIs to synchronize the Workflow directory service with Oracle Internet Directory if you implement Oracle Internet Directory integration. One event is raised for each changed user. See: Synchronizing Workflow Directory Services with Oracle Internet Directory, *Oracle Workflow Administrator's Guide* and Workflow LDAP APIs, *Oracle Workflow API Reference*.

Internal Name
oracle.apps.global.user.change

Status
Enabled

Generate Function
wf_entity_mgr.gen_xml_payload

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

The standalone version of Oracle Workflow provides two default subscriptions to the User Entry Has Changed event. The first subscription creates a new user record in the WF_LOCAL_ROLES table for the standalone version of Oracle Workflow when the User Entry Has Changed event is raised locally, if that user does not already exist in the

table. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.global.user.change
Phase	50
Status	Enabled
Rule Data	Key
Rule Function	wf_sso.user_create_rf
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The second subscription loads the changed user data into the WF_LOCAL_ROLES table for the standalone version of Oracle Workflow when the User Entry Has Changed event is raised locally. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.global.user.change
Phase	60
Status	Enabled
Rule Data	Key
Rule Function	wf_sso.user_change
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

In Oracle Applications, the embedded version of Oracle Workflow provides one default subscription to the User Entry Has Changed event that loads the changed user data into the WF_LOCAL_ROLES table for Oracle Applications when the User Entry Has

Changed event is raised locally. You should enable this subscription if you are using the version of Oracle Workflow embedded in Oracle Applications and you want to implement integration with Oracle Internet Directory. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.global.user.change
Phase	51
Status	Disabled
Rule Data	Key
Rule Function	fnd_user_pkg.user_create_rf
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Notification Events

Event for Notification Send

Oracle Workflow uses this event to send outbound notification messages.

Internal Name
oracle.apps.wf.notification.send

Status
Enabled

Generate Function
WF_XML.Generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides one default subscription to the Notification Send event, which is the default subscription if a notification mailer encounters errors during the generation and sending of outbound messages. When the Notification Send event is received from an Error source (that is, when it is dequeued from the WF_ERROR or WF_JAVA_ERROR queue), this subscription sends the event message to the Default Event Error process in the System: Error item type. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Error
Event Filter	oracle.apps.fnd.wf.notification.send
Phase	50
Status	Enabled
Rule Data	Key
Rule Function	wf_xml.error_rule
Workflow Item Type	WFERROR
Workflow Process Name	DEFAULT_EVENT_ERROR
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Event for Notification Reassign

Oracle Workflow uses this event to send an outbound notification message to the new recipient when a notification is reassigned.

Internal Name
oracle.apps.wf.notification.reassign

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Event for Notification Cancel

Oracle Workflow uses this event to send an outbound cancellation message when a previously sent notification is canceled.

Internal Name
oracle.apps.wf.notification.cancel

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Notification Send Group

This event group contains events for various types of outbound messages sent by notification mailers.

Internal Name
oracle.apps.wf.notification.send.group

Status
Enabled

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Members

- oracle.apps.wf.notification.send.
- oracle.apps.wf.notification.reassign
- oracle.apps.wf.notification.cancel

Oracle Workflow provides two default subscriptions to the Notification Send Group. The first subscription places the event message on the WF_NOTIFICATION_OUT agent to be sent by a notification mailer when one of the group member events is raised locally. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.fnd.wf.notification.send.group
Phase	100
Status	Enabled
Rule Data	Message
Rule Function	wf_rule.default_rule
Out Agent	WF_NOTIFICATION_OUT@*<local system>*
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The second subscription can be used to place the event message on the WF_SMTP_O_1_QUEUE to be processed by the C-based version of the Notification Mailer, when one of the group member events is raised locally. This subscription should only be used when Oracle Support determines it to be necessary. Do not enable this subscription unless you are directed to do so by Oracle Support. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.wf.notification.send.group
Phase	0
Status	Disabled
Rule Data	Key
Rule Function	wf_xml.sendnotification
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Send Summary Notification Event

Oracle Workflow raises this event to generate and send a summary notification for each user and role with a notification preference of `SUMMARY` or `SUMHTML` when a Launch Summary Notifications event is raised.

Internal Name
oracle.apps.wf.notification.summary.send

Status
Enabled

Generate Function
WF_XML.Generate

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides one default subscription to the Send Summary Notification event. When the Send Summary Notification event is raised locally, this subscription places the event message on the WF_NOTIFICATION_OUT agent to be sent by a notification mailer. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.fnd.wf.notification.summary.send
Phase	0
Status	Enabled
Rule Data	Message
Rule Function	wf_rule.default_rule
Out Agent	WF_NOTIFICATION_OUT@*<local system>*
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Notification Send Error Event

This event is not currently used.

Internal Name
oracle.apps.wf.notification.send.error

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Receipt of Incoming Response Event

Oracle Workflow raises this event when an inbound message is received by a notification mailer.

Internal Name
oracle.apps.wf.notification.receive.message

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Return to Sender Event

Oracle Workflow raises this event if there are problems during the processing of an inbound response message received by a notification mailer, and these problems require the original notification or the response to be resent.

Internal Name
oracle.apps.wf.notification.receive.sendreturn

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Notification Recipient Is Unavailable Event

Oracle Workflow raises this event if an inbound response message received by a notification mailer contains a string of text that matches a tag pattern associated with the Unavailable tag action.

Internal Name
oracle.apps.wf.notification.receive.unavail

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Notification Error Event

Oracle Workflow raises this event if an inbound response message received by a notification mailer contains a string of text that matches a tag pattern associated with the Error tag action.

Internal Name
oracle.apps.wf.notification.receive.error

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Notification Receive Group

This event group contains events for various types of incoming messages received by notification mailers.

Internal Name
oracle.apps.wf.notification.receive

Status
Enabled

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Members

- oracle.apps.wf.notification.receive.message
- oracle.apps.wf.notification.receive.sendreturn
- oracle.apps.wf.notification.receive.unavail
- oracle.apps.wf.notification.receive.error

Oracle Workflow provides one default subscription to the Notification Receive Group. This subscription processes the incoming message when one of the group member events is received from an external source.

- oracle.apps.wf.notification.receive.message - The notification mailer performs normal response processing for the message. See: Inbound Notification Mailer Processing, *Oracle Workflow Administrator's Guide*.
- oracle.apps.wf.notification.receive.sendreturn - The outbound notification is resent.
- oracle.apps.wf.notification.receive.unavail - The message is moved to the discard folder, and the notification mailer continues waiting for a reply to the notification since the notification's status is still `OPEN`. However, the notification's mail status is updated to `UNAVAIL`.
- oracle.apps.wf.notification.receive.error - The message is moved to the discard folder, and Oracle Workflow initiates an error process, if one is defined for the workflow process to which the notification activity belongs. The notification's status is still `OPEN`, but its mail status and activity status are updated to `ERROR`. Ideally, the workflow administrator corrects the problem and resends the notification by updating its mail status to `MAIL`.

This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	oracle.apps.wf.notification.receive
Phase	0
Status	Enabled
Rule Data	Message
Rule Function	wf_xml.receive
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Event for Notification Respond

Oracle Workflow raises this event whenever a user responds to a notification.

Internal Name
oracle.apps.wf.notification.respond

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Event for Notification Close

Oracle Workflow raises this event whenever a notification is closed.

Internal Name
oracle.apps.wf.notification.close

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Notification Mailer Events

Launch Summary Notifications

Oracle Workflow uses this event to make a notification mailer service component send summary notifications to roles with a notification preference of `SUMMARY` or `SUMHTML`. A summary notification lists all notifications for a role that are open at the time that the summary is sent.

Internal Name
oracle.apps.fnd.wf.mailer.Mailer.notification.summary

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides one default subscription to the Launch Summary Notifications event. When the Launch Summary Notifications event is raised locally, this subscription raises the oracle.apps.wf.notification.summary.send event for each role with a notification preference of `SUMMARY` or `SUMHTML`. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.fnd.wf.mailer.Mailer.notification.summary
Phase	0
Status	Enabled
Rule Data	Message
Rule Function	wf_xml.summaryrule
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Unsolicited Email Threshold Reached Event

A notification mailer raises this event when it receives a second unsolicited message from an e-mail address to which it has already sent a warning message for unsolicited mail.

> **Note:** If the mailer node receives additional unsolicited messages from the same e-mail address, those messages are simply discarded. The mailer does not raise the Unsolicited Email Threshold Reached event again.

Internal Name
oracle.apps.wf.mailer.unsolicited

Status
Disabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Business Event System Control Events

Business Event System Control Ping Event

The Workflow Control Queue Cleanup program raises this event to send to the middle tier subscribers for the WF_CONTROL queue, in order to check whether they are still active.

Internal Name
oracle.apps.wf.bes.control.ping

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Business Event System Control Group

This event group contains control events for Oracle Applications or standalone Oracle Workflow internal processing. Events in this group are meant to be placed on the WF_CONTROL queue for middle tier processes to dequeue.

Internal Name
oracle.apps.wf.bes.control.group

Status
Enabled

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Members
oracle.apps.wf.bes.control.ping

Oracle Workflow provides one default subscription to the Business Event System Control Group. This subscription places the event message on the standard WF_CONTROL queue when one of the group member events is raised locally. Middle tier subscribers to this queue can then dequeue the event message. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.wf.bes.control.group
Phase	50
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.default_rule
Out Agent	WF_CONTROL@*<local system>*
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Business Event System Applications Control Group

This event group contains control events for Oracle Applications internal processing. Events in this group are meant to be updated with Oracle Applications context information and placed on the WF_CONTROL queue for middle tier processes to dequeue. This group does not currently have any seeded members.

Internal Name
oracle.apps.fnd.bes.control.group

Status
Enabled

Owner Name
Application Object Library

Owner Tag
FND

Customization Level
Limit

Oracle Application Object Library provides one default subscription to the Business Event System Applications Control Group. When one of the group member events is raised locally, this subscription updates the event message with Oracle Applications context data such as the database ID and security group ID and then places the event message on the standard WF_CONTROL queue. Middle tier subscribers to this queue can then dequeue the event message. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.fnd.bes.control.group
Phase	50
Status	Enabled
Rule Data	Key
Rule Function	fnd_bes_proc.process_event
Out Agent	WF_CONTROL@*<local system>*
Priority	Normal
Owner Name	Application Object Library
Owner Tag	FND
Customization Level	Limit

Related Topics

Cleaning Up the Workflow Control Queue, *Oracle Workflow Administrator's Guide*

Business Event System Cleanup API, *Oracle Workflow API Reference*

Generic Service Component Framework Control Events

Start Event

The Generic Service Component Framework uses this event to start a service component.

Internal Name
oracle.apps.fnd.cp.gsc.SvcComponent.start

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Core

Refresh Event

The Generic Service Component Framework uses this event to refresh a running service component with new parameter values, for those parameters that can be dynamically refreshed.

Internal Name
oracle.apps.fnd.cp.gsc.SvcComponent.refresh

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Core

Suspend Event

The Generic Service Component Framework uses this event to suspend processing for a running service component.

Internal Name
oracle.apps.fnd.cp.gsc.SvcComponent.suspend

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Core

Resume Event

The Generic Service Component Framework uses this event to resume processing for a suspended service component.

Internal Name
oracle.apps.fnd.cp.gsc.SvcComponent.resume

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Core

Stop Event

The Generic Service Component Framework uses this event to stop a running service component.

Internal Name
oracle.apps.fnd.cp.gsc.SvcComponent.stop

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Core

GSC Business Event System Control Group

This event group contains control events for Generic Service Component Framework internal processing. Events in this group are meant to be placed on the WF_CONTROL queue for middle tier processes to dequeue.

Internal Name
oracle.apps.fnd.cp.gsc.bes.control.group

Status
Enabled

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Core

Members

- oracle.apps.fnd.cp.gsc.SvcComponent.start
- oracle.apps.fnd.cp.gsc.SvcComponent.refresh
- oracle.apps.fnd.cp.gsc.SvcComponent.suspend
- oracle.apps.fnd.cp.gsc.SvcComponent.resume
- oracle.apps.fnd.cp.gsc.SvcComponent.stop

Oracle Workflow provides one default subscription to the GSC Business Event System Control Group. This subscription places the event message on the standard

WF_CONTROL queue when one of the group member events is raised locally. Middle tier subscribers to this queue can then dequeue the event message. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.fnd.cp.gsc.bes.control.group
Phase	50
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.error_rule
Out Agent	WF_CONTROL@*<local system>*
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Core

For more information about the Generic Service Component Framework, please refer to the Oracle Applications Manager online help or the Oracle Enterprise Manager online help.

Workflow Engine Events

oracle.apps.wf.engine.abort Event

The Oracle Workflow Engine raises this event when a workflow process is aborted using *WF_ENGINE.AbortProcess*. See: AbortProcess, *Oracle Workflow API Reference*.

Internal Name
oracle.apps.wf.engine.abort

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Core

oracle.apps.wf.engine.retry Event

The Oracle Workflow Engine raises this event when an activity is retried using *WF_ENGINE.HandleError*. See: HandleError, *Oracle Workflow API Reference*.

Internal Name
oracle.apps.wf.engine.retry

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Core

oracle.apps.wf.engine.skip Event

The Oracle Workflow Engine raises this event when an activity is skipped using *WF_ENGINE.HandleError*. See: HandleError, *Oracle Workflow API Reference*.

Internal Name
oracle.apps.wf.engine.skip

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Core

Directory Service Events

Workflow User Updated Event

Oracle Workflow raises this event whenever a user is updated in the Oracle Workflow directory service.

Internal Name
oracle.apps.fnd.wf.ds.user.updated

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides one default subscription to the Workflow User Updated event. When the Workflow User Updated event is raised locally, this subscription makes corresponding updates to any associations of that user with roles. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.fnd.wf.ds.user.updated
Phase	100
Status	Enabled
Rule Data	Key
Rule Function	WF_ROLE_HIERARCHY.Denormalize_User_Role_RF
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Workflow Role Updated Event

Oracle Workflow raises this event whenever a role is updated in the Oracle Workflow directory service.

Internal Name
oracle.apps.fnd.wf.ds.role.updated

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides one default subscription to the Workflow Role Updated event. When the Workflow Role Updated event is raised locally, this subscription makes corresponding updates to any associations of that role with users. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.fnd.wf.ds.role.updated
Phase	100
Status	Enabled
Rule Data	Key
Rule Function	WF_ROLE_HIERARCHY.Denormalize_User_Role_RF
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Workflow User/Role Relationship Created Event

Oracle Workflow raises this event whenever an association of a user with a role is created in the Oracle Workflow directory service.

Internal Name
oracle.apps.fnd.wf.ds.userRole.created

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides two default subscriptions to the Workflow User/Role Relationship Created event. The first subscription updates the hierarchy of role assignments for a user when the Workflow User/Role Relationship Created event is raised locally. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.fnd.wf.ds.userRole.created
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	WF_ROLE_HIERARCHY.Cascade_RF
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The second subscription stores denormalized associations of the user with roles, based on the hierarchy of role assignments for the user, when the Workflow User/Role Relationship Created event is raised locally. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.fnd.wf.ds.userRole.created
Phase	100
Status	Enabled
Rule Data	Key
Rule Function	WF_ROLE_HIERARCHY.Denormalize_User_Role_RF
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Workflow User/Role Relationship Updated Event

Oracle Workflow raises this event whenever an association of a user with a role is updated in the Oracle Workflow directory service.

Internal Name
oracle.apps.fnd.wf.ds.userRole.updated

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides one default subscription to the Workflow User/Role Relationship Updated event. This subscription updates the hierarchy of role assignments for a user when the Workflow User/Role Relationship Updated event is raised locally. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.fnd.wf.ds.userRole.updated
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	WF_ROLE_HIERARCHY.Cascade_RF
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Workflow Role Hierarchy Relationship Created Event

Oracle Workflow raises this event whenever a hierarchical relationship of a role to another role is created in the Oracle Workflow directory service.

Internal Name
oracle.apps.fnd.wf.ds.roleHierarchy.relationshipCreated

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides one default subscription to the Workflow Role Hierarchy Relationship Created event. This subscription runs the Workflow Role Hierarchy Propagation concurrent program (`FNDWFDSRHP`) to update the denormalized associations of users with roles, based on the hierarchy of roles, when the Workflow Role Hierarchy Relationship Created event is raised locally. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.fnd.wf.ds.roleHierarchy.relationshipCreated
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	WF_ROLE_HIERARCHY.Propagate_RF
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Workflow Role Hierarchy Relationship Updated Event

Oracle Workflow raises this event whenever a hierarchical relationship of a role to another role is updated in the Oracle Workflow directory service.

Internal Name
oracle.apps.fnd.wf.ds.roleHierarchy.relationshipUpdated

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides one default subscription to the Workflow Role Hierarchy Relationship Updated event. This subscription runs the Workflow Role Hierarchy Propagation concurrent program (`FNDWFDSRHP`) to update the denormalized associations of users with roles, based on the hierarchy of roles, when the Workflow Role Hierarchy Relationship Updated event is raised locally. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.fnd.wf.ds.roleHierarchy.relationshipUpdated
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	WF_ROLE_HIERARCHY.Propagate_RF
Priority	Normal
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Workflow Send Protocol

The Workflow Send Protocol process is a sample workflow process that demonstrates receiving, sending, and acknowledging event messages. Depending on your requirements, you can copy or customize this process to accommodate your organization's specific needs.

The Workflow Send Protocol process receives an event message from a subscription, sends the event message to the inbound agent specified in the subscription, waits to receive an acknowledgement if required, and also sends an acknowledgement if required. You can use the process on one system to send a message to another system, for example, and you can also use the same process on the second system to send the acknowledgement message back to the first system.

The Workflow Send Protocol workflow consists of one process, the Workflow Event Protocol process. This process is launched when it receives an event message from an event subscription.

You can start the Workflow Send Protocol workflow by any of the following methods:

- Raise the Workflow Send Protocol event from the Raise event page. Enter a unique event key, and enter any valid XML document as the event data. A predefined subscription sends the event message to the Workflow Event Protocol process. See: Raising Events, page 8-44 and Workflow Send Protocol Events, page 9-51.
- The process can also be started when an agent receives the Workflow Send Protocol event from an external source. A predefined subscription sends the event message to the Workflow Event Protocol process. See: Workflow Send Protocol Events, page 9-51.
- Define your own subscription to send any event you choose to the Workflow Event Protocol process. In the subscription, specify the workflow item type as `WFSNDPRT`

and the workflow process name as `WFEVPRTC`. Ensure that you either use the default rule function or include send processing in your custom rule function to send the event to the workflow. The Workflow Event Protocol process starts when the subscription is executed and the process receives the event message. See: To Define an Event Subscription, page 8-38.

Related Topics

The Workflow Send Protocol Item Type, page 9-46

Summary of the Workflow Event Protocol Process, page 9-47

Workflow Event Protocol Process Activities, page 9-49

Workflow Send Protocol Events, page 9-51

The Workflow Send Protocol Item Type

The Workflow Send Protocol process is associated with an item type called Workflow Send Protocol. Currently there is one workflow process associated with Workflow Send Protocol: the Workflow Event Protocol process.

To view the details of the Workflow Send Protocol item type in the Workflow Builder, choose Open from the File menu. Then connect to the database and select the Workflow Send Protocol item type, or connect to a file called `wfsndprt.wft` in the *<ORACLE_HOME>*`\wf\Data\`*<language>* directory.

If you examine the property page of Workflow Send Protocol, you see that it has a persistence type of Temporary and persistence number of days of 0. This means that the runtime data associated with any work items for this item type are eligible for purging as soon as they complete.

The Workflow Send Protocol item type also has several attributes associated with it. These attributes reference information in the Workflow application tables. The attributes are used and maintained by function activities as well as event activities throughout the process. The following table lists the Workflow Send Protocol item type attributes.

Workflow Send Protocol Item Type Attributes

Display Name	Description	Type	Length/Format/ Lookup Type
Event Name	The internal name of the event	Text	
Event Key	The event key that uniquely identifies the specific instance of the event	Text	
Event Message	The event message	Event	
To Agent	The inbound agent that receives the event message, in the format *<agent>@<system>*	Text	
From Agent	The outbound agent that sends the event message, in the format *<agent>@<system>*	Text	
Acknowledge Required?	An option that specifies whether the event message that is sent requires an acknowledgement from the recipient	Text	
Send Acknowledgement?	An option that specifies whether to send an acknowledgement of a message that is received	Text	
Acknowledge Message	The acknowledgement message that is sent	Event	
Acknowledge To Agent	The inbound agent that receives the acknowledgement message, in the format *<agent>@<system>*	Text	
Subscription GUID	The globally unique identifier of the subscription	Text	

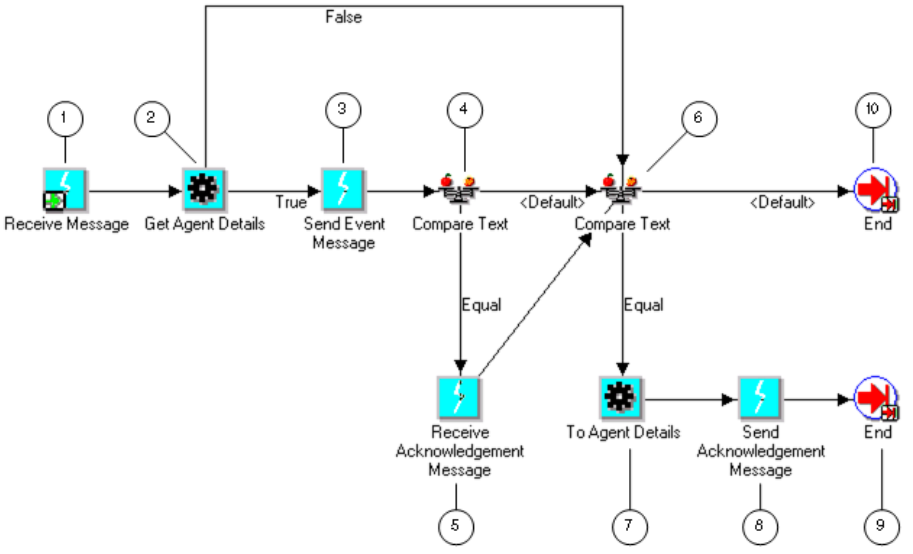
Summary of the Workflow Event Protocol Process

To view the properties of the Workflow Event Protocol process, select the process in the navigator tree, and then choose Properties from the Edit menu. This process activity is runnable, indicating that it can be initiated as a top level process to run.

When you display the Process window for the Workflow Event Protocol process, you see that the process consists of eight unique activities, some of which are reused to make up the ten activity nodes that appear in the workflow diagram. To examine the activities of

the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Workflow Event Protocol Process



The Workflow Send Protocol workflow begins when the Event Manager sends an event message to the Workflow Event Protocol process. For example, when you raise the Workflow Send Protocol event locally or receive that event from an external source, predefined subscriptions send the event message to the Workflow Event Protocol process. See: Workflow Send Protocol Events, page 9-51.

The workflow begins at node 1 with the Receive message activity. At node 2, the process attempts to retrieve the agent details for the intended outbound and inbound agents from the subscription.

If no inbound agent is specified, the process continues immediately to node 6 to determine whether to send an acknowledgement message.

If the subscription does specify a To Agent, the process sends the event message to that agent. Then the process determines whether the event message requires an acknowledgement from the recipient, based on a subscription parameter. If an acknowledgement is required, the Workflow Engine waits to receive the acknowledgement message. Otherwise, the process continues immediately to node 6 to determine whether to send an acknowledgement message.

At node 6, the process determines whether it should send an acknowledgement of the original message that it received. If no acknowledgement needs to be sent, the process ends at this point. Otherwise, the process retrieves the agent details for the inbound agent where the acknowledgement must be sent and sends the acknowledgement message to that agent. Then the process ends.

Workflow Event Protocol Process Activities

Following is a description of each activity in the process, listed by the activity's display name.

Receive Message (Node 1)

This event activity receives the event message that is sent to the Workflow Event Protocol process by the Event Manager to start a new item.

Event Action
Receive

Event Filter
None

Prerequisite Activities
None

Item Attributes Set by Activity
Event Name, Event Key, Event Message

Get Agent Details (Node 2)

This function activity attempts to retrieve the agent details from the subscription for the outbound agent that should send the message and the inbound agent that should receive the message. If no inbound agent is specified, the process continues immediately to node 6. If an inbound agent is specified, but no outbound agent is specified, the function selects a default outbound agent on the local system.

Function
WF_STANDARD.GETAGENTS

Result Type
Boolean

Prerequisite Activities
Receive Message

Item Attributes Retrieved by Function
Subscription GUID

Item Attributes Set by Function
To Agent, From Agent

Send Event Message (Node 3)

This event activity sends the event message from an outbound agent on the local system to the specified inbound agent.

Event Action
Send

Prerequisite Activities
Get Agent Details

Item Attributes Retrieved by Activity
Event Message, From Agent, To Agent

Compare Text (Node 4)

This Standard function activity compares two text values. At this node, the process checks the Acknowledge Required? item attribute to determine whether the event message sent at node 3 requires an acknowledgement from the recipient.

If the subscription that initiated the process included the parameter name and value pair `ACKREQ=Y`, the Workflow Engine sets the Acknowledge Required? item attribute to `Y` when the process is launched. In this case, the process continues from node 4 to node 5. Otherwise, the process continues directly from node 4 to node 6.

Function
WF_STANDARD.COMPARE

Result Type
Comparison

Prerequisite Activities
None

Item Attributes Retrieved by Activity
Acknowledge Required?

Receive Acknowledgement Message (Node 5)

This event activity waits to receive the Workflow Send Protocol Acknowledgement event message that is returned to the Workflow Event Protocol process from the system that received the event message sent at node 3.

Event Action
Receive

Event Filter
oracle.apps.wf.event.wf.ack

Prerequisite Activities
Compare Text

Compare Text (Node 6)

This Standard function activity compares two text values. At this node, the process checks the Send Acknowledgement? item attribute to determine whether to send an acknowledgement of the original message that it received.

If the original message requires an acknowledgement, the Workflow Engine sets the Send Acknowledgement? item attribute to `Y` when the process is launched. In this case, the process continues from node 6 to node 7. Otherwise, the process ends at node 10.

Function
WF_STANDARD.COMPARE

Result Type
Comparison

Prerequisite Activities
None

Item Attributes Retrieved by Activity
Send Acknowledgement?

To Agent Details (Node 7)

This function activity selects an inbound agent on the originating system where the acknowledgement must be sent and retrieves the agent details for that agent.

Function
WF_STANDARD.GETACKAGENT

Result Type
None

Prerequisite Activities
Compare Text

Item Attributes Set by Function
Acknowledge To Agent

Send Acknowledgement Message (Node 8)

This event activity sends the Workflow Send Protocol Acknowledgement message from an outbound agent on the local system to the inbound agent identified at node 7.

Event Action
Send

Prerequisite Activities
To Agent Details

Item Attributes Retrieved by Activity
Acknowledge Message, Event Key, Acknowledge To Agent

End (Nodes 9 and 10)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

Workflow Send Protocol Events

Workflow Send Protocol Event

You can raise this event from the Raise Event page to send the event message to an agent using the Workflow Event Protocol process in the Workflow Send Protocol item type. This workflow process lets you specify whether the message

requires an acknowledgement from the recipient, and whether you want to send an acknowledgement of a message that you have received.

Internal Name
oracle.apps.wf.event.wf.send

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides two default subscriptions to the Workflow Send Protocol event. The first subscription sends the event message to the Workflow Event Protocol process in the Workflow Send Protocol item type when the Workflow Send Protocol event is raised locally. A subscription parameter specifies that the message requires an acknowledgement. This subscription is enabled by default. You can also create a copy of this subscription with the same properties, except you should give your new subscription a customization level of User, and add an outbound agent and inbound agent to the subscription to specify where you want the Workflow Event Protocol process to send the event message. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	oracle.apps.wf.event.wf.send
Phase	50
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.workflow_protocol
Workflow Item Type	WFSNDPRT
Workflow Process Name	WFEVPRTC
Parameters	ACKREQ=Y
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The second subscription sends the event message to the Workflow Event Protocol process in the Workflow Send Protocol item type when the Workflow Send Protocol event is received from an external source. This subscription is enabled by default. You can optionally create a copy of this subscription with the same properties, except you should give your new subscription a customization level of User, and add an outbound agent and inbound agent to the subscription to specify that you want the Workflow Event Protocol process to send the event message on to another agent. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	oracle.apps.wf.event.wf.send
Phase	50
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.workflow_protocol
Workflow Item Type	WFSNDPRT
Workflow Process Name	WFEVPRTC
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

Workflow Send Protocol Acknowledgement Event

Oracle Workflow sends this event back to the originating system when an event message sent to or from the Workflow Event Protocol process requires an acknowledgement.

Internal Name
oracle.apps.wf.event.wf.ack

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides one default subscription to the Workflow Send Protocol Acknowledgement Event. This subscription sends the event message to the Workflow Event Protocol process in the Workflow Send Protocol item type when the Workflow Send Protocol Acknowledgement event is received from an external source. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	oracle.apps.wf.event.wf.ack
Phase	50
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.workflow_protocol
Workflow Item Type	WFSNDPRT
Workflow Process Name	WFEVPRTC
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

10

Demonstration Workflow Processes

This chapter describes the demonstration workflow processes provided with Oracle Workflow. These demonstration processes showcase many Oracle Workflow features.

This chapter covers the following topics:

- Sample Workflow Processes
- Displaying the Process Diagram of a Sample Workflow
- Requisition Process
- Initiating the Requisition Workflow
- The Requisition Item Type
- Event System Demonstration
- Initiating the Event System Demonstration Workflow
- The Event System Demonstration Item Type

Sample Workflow Processes

The following sample workflow processes are included with Oracle Workflow. Each of these processes illustrates the integration of different Oracle Workflow features. You can use these processes to verify your installation of Oracle Workflow.

- Requisition Process - illustrates results-based branching, parallel branching, subprocesses, timeouts, looping, and integration of PL/SQL documents in a notification. See: Requisition Process, page 10-3.
- Event System Demonstration Processes - illustrate sending and receiving business events between two systems and using external Java function activities. See: Event System Demonstration, page 10-28.

You can set up and initiate these sample processes if you are using the standalone version of Oracle Workflow. If you are using Oracle Workflow embedded in Oracle Applications, you should consider these processes mainly as examples for explanation purposes and not for demonstration use. The files necessary to set up and run these demonstrations are not provided with the version of Oracle Workflow embedded in Oracle Applications.

You can initiate these sample workflows from the Workflow Demonstrations home page or the Launch Processes Web page. You can access the Workflow Demonstrations home page using the URL:

`<webagent>/wf_demo.home`

> **Note:** You can also access the Workflow Demonstrations Web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page, *Oracle Workflow Administrator's Guide*.

The Workflow Demonstrations home page displays your notifications Worklist in the right-hand frame and in the left-hand frame lists links that let you initiate each of the sample workflows from a different Web page.

Related Topics

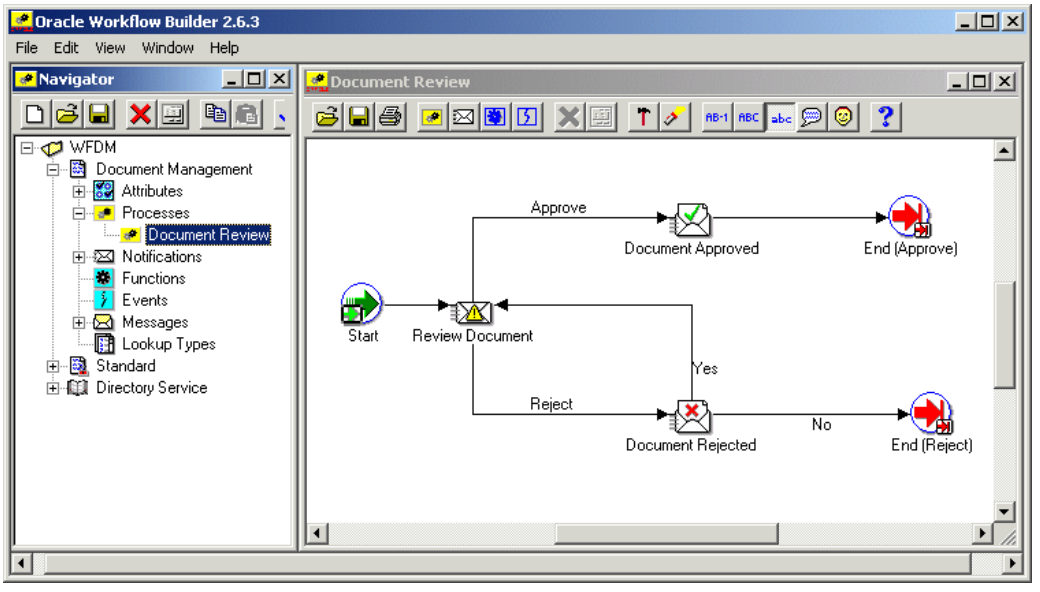
Testing Workflow Definitions, page 7-1

Displaying the Process Diagram of a Sample Workflow, page 10-2

Displaying the Process Diagram of a Sample Workflow

You can view the process diagram of a sample process in the Process window of Oracle Workflow Builder.

Sample Process Diagram



To Display a Sample Process in Oracle Workflow Builder:

1. Choose Open from the File menu. Connect to your Oracle Workflow database and select the item type you want.

   Alternatively, you can connect to any of the sample workflow definition files, located in the `ORACLE_HOME\wf\data\<language>` directory on your PC.

2. Expand the data source, then the item type branch within that data source.

3. Expand the Processes branch and then double-click on a process activity to display the process diagram in a Process window.

Requisition Process

The Requisition process is an example of a workflow process that is initiated when you create a new requisition to purchase an item. The Requisition process is based on two tables that store approval hierarchy and spending authority information.

When you submit a requisition in this demonstration, the process sends a notification to the next manager in the approval hierarchy to approve the requisition. If the spending limit of the approving manager is less than the requisition amount, the process forwards the requisition to the next higher manager in the approval hierarchy until it finds a manager with the appropriate spending limit to approve the requisition. Each intermediate manager must approve the requisition to move it to the next higher manager. Once a manager with the appropriate spending limit approves the requisition, the process ends with a result of Approve.

The process can end with a result of rejected if:

- Any manager rejects the requisition.
- The requisition amount is greater than the highest spending limit.
- The requisition's requestor does not have a manager.

You can set up and initiate this example process if you are using the standalone version of Oracle Workflow. If you are using Oracle Workflow embedded in Oracle Applications, you should consider this process mainly as an example for explanation purposes and not for demonstration use. The files necessary to set up and run this demonstration are not provided with the version of Oracle Workflow embedded in Oracle Applications.

> **Important:** For detailed information about runnable workflow processes that are integrated with Oracle Applications, refer to the appropriate Oracle Applications user's guide or online documentation. See: Predefined Workflows Embedded in Oracle E-Business Suite, page C-1.

> **Important:** Oracle Self-Service Web Applications provides a predefined Requisition Process that is different from the version of the example process documented here. The example process documented in this section is for demonstration purposes only and not for production use.

This sample workflow is based on the demonstration data model. The data model includes two tables with data: one table maintains an employee approval hierarchy and the other maintains the spending limit of each employee. These two tables make up the database application that we use to approve a requisition. In addition, the data model also includes a directory service that identifies the Oracle Workflow users and roles in this sample implementation.

There are two ways you can initiate the Requisition process based on a fictitious requisition: run a script or submit a requisition using a Web-based interface. Both methods require that you provide the name of the employee who prepared the requisition, the requisition amount, the requisition number, a requisition description, a requisition process owner and the name of the workflow process to initiate.

This section describes the Requisition process in detail to give you an understanding of what each activity in this workflow accomplishes.

Related Topics

Installing the Requisition Data Model, page 10-4

Initiating the Requisition Workflow, page 10-6

The Requisition Item Type, page 10-8

Summary of the Requisition Approval Process, page 10-9

Requisition Process Activities, page 10-11

Summary of the Notify Approver Subprocess, page 10-15

Notify Approver Subprocess Activities, page 10-16

Sample StartProcess Function, page 10-18

Example Function Activities, page 10-21

Example Notification Activity, page 10-26

Installing the Requisition Data Model

The Requisition data model is installed only with the standalone version of Oracle Workflow. The data model is automatically installed for you by the Workflow Configuration Assistant. The files used in the installation are copied to the `ORACLE_HOME/wf/demo` and `ORACLE_HOME/wf/<language>` directories in your Oracle Workflow server directory structure.

> **Important:** For the Requisition process demonstration to work properly, you should perform the steps required to set up Oracle Workflow after the installation. See: Overview of Setting Up, *Oracle Workflow Administrator's Guide*.

The installation does the following:

- Calls the script `wfdemou.sql` to create a database account for each of the users listed in the seed data table shown below. The script creates public grants and synonyms so that these accounts have full access to Oracle Workflow's Web-based user interface.

  > **Important:** For security reasons, the installation process automatically locks these user accounts after they are created. Before you can begin using the accounts, you must unlock them using a script called `wfdemoul.sql`. This script is located in the `ORACLE_HOME/wf/demo` directory. Connect to the `SYSTEM` database account using SQL*Plus and run the script using the following command:
  >
  > ```
  > sqlplus SYSTEM/<SYSTEM pwd> @wfdemoul
  > ```
  >
  > See your Oracle DBA if you need more information about the `SYSTEM` account and password.

- Calls a script called `wfdemoc.sql` to create two tables with seed data. These tables make up the demonstration database application that is workflow-enabled:
  - WF_REQDEMO_EMP_HIERARCHY - maintains the employee approval hierarchy. The approval chain consists of these employee user IDs listed

in ascending order with the employee having the most authority listed last: `BLEWIS`, `KWALKER`, `CDOUGLAS`, and `SPIERSON`.

- WF_REQDEMO_EMP_AUTHORITY - maintains the spending limit for each employee. The limit for each employee follows the employee's user ID: `BLEWIS`:500, `KWALKER`:1000, `CDOUGLAS`:2000, and `SPIERSON`:3000.

- The script `wfdemoc.sql` also inserts seed data into the WF_LOCAL_ROLES and WF_LOCAL_USER_ROLES tables. The following table shows the users and roles that are seeded by the script.

Demonstration Users and Roles

User	ADMIN Role	MANAGERS Role	WORKERS Role	OTHERS Role
SYSADMIN	yes			
WFADMIN	yes			
BLEWIS			yes	
KWALKER			yes	
CDOUGLAS			yes	yes
SPIERSON		yes		yes

Important: Each user has an e-mail address of `'WFINVALID'` and each role has an e-mail address identical to its role name. You can change the users' and roles' e-mail addresses to other values by calling the Directory Service APIs *SetAdHocUserAttr* or *SetAdHocRoleAttr*. Alternatively, if you want e-mail notifications for all the users and roles to go to a single e-mail inbox, you can specify a test e-mail address in the Test Address configuration parameter when you configure a notification mailer. For more information, please refer to the Oracle Workflow Manager sections of the Oracle Applications Manager online help for Oracle Workflow embedded in Oracle Applications, or to the Oracle Workflow Manager sections of the Oracle Enterprise Manager online help for standalone Oracle Workflow.

Important: Also all users except `BLEWIS` have a Notification Preference of '`MAILHTML`', which allows them, in addition to viewing notifications from the Notifications Web page, to get individual notifications via e-mail. `BLEWIS` has a Notification Preference of '`SUMMARY`', which allows him, in addition to viewing notifications from the Notifications Web page, to receive a periodic e-mail summarizing all his currently open notifications. Note that a notification mailer must be set up to deliver e-mail notifications.

Important: Your Oracle Workflow directory service views must map to the WF_LOCAL_ROLES and WF_LOCAL_USER_ROLES tables to include the users and roles of the Requisition data

model. See: Setting Up an Oracle Workflow Directory Service, *Oracle Workflow Administrator's Guide*.

- Calls the scripts `wfdemos.sql` and `wfdemob.sql` to create the PL/SQL spec and body for packages called WF_REQDEMO and WF_DEMO. These packages contain:
  - The PL/SQL stored procedures associated with the demonstration home page.
  - The PL/SQL stored procedures called by the function activities used in the Requisition Process workflow.
  - The PL/SQL procedure WF_REQDEMO.Create_Req called by the Oracle Workflow Web agent to generate the Web-based interface page for the Requisition process demonstration.
- Runs the Workflow Resource Generator to load messages from `wfdemo.msg` into the database. The messages are used by the Web-based interface page for the Requisition process demonstration.
- Loads the Requisition Process workflow definition from `wfdemo.wft` into the database. You can view this process in Oracle Workflow Builder.

Initiating the Requisition Workflow

You can use any of the following methods to initiate the Requisition workflow:

- Run the script `wfrund.sql`., page 10-6
- Access the Requisition Demonstration Web page from the Workflow Demonstrations home page., page 10-7
- Use the Launch Processes Web page. See: Testing Workflow Definitions, page 7-1.

You can also create your own custom end-user application interface to let users create requisitions that automatically initiate the Requisition process workflow. You must, however, customize the application interface such that when a user saves the requisition to the application database, the application calls a PL/SQL stored procedure similar to *WF_REQDEMO.StartProcess* that initiates the Requisition process. See: Sample StartProcess Function, page 10-18.

To Run wfrund.sql:

1. Enter the following command to run the script `wfrund.sql` in SQL*Plus:

   ```
   sqlplus <username>/<password>@<alias> @wfrund.sql
   <req_num> <req_desc> <req_amount> <requestor> <req_process_owner>
   <process_int_name> <item_type>
   ```

   Replace *<username>/<password>@<alias>* with the username, password, and alias for the database account where you installed the demonstration data model.

   Replace *<req_num>* with the requisition number that uniquely identifies the requisition.

   > **Note:** The requisition number is used as the item key for the process, so it must contain only single-byte characters.

Replace *<req_desc>* with an end-user defined description that uniquely identifies the requisition.

Replace *<req_amount>* with the amount of the requisition, *<requestor>* with the name of the requisition requestor (who should be listed in the employee approval hierarchy), *<req_process_owner>* with the name of the requisition process owner (who should be listed in the employee approval hierarchy), *<process_int_name>* with the internal name of the process activity (in this case, `REQUISITION_APPROVAL`) and *<item_type>* with the internal name of the item type that the workflow process is associated with.

2. When this script completes, enter `Commit` at the `SQL>` prompt to save the transaction before quitting from SQL*Plus.
3. Based on the approval hierarchy, you can either log on as the requisition requestor or the requestor's manager to follow and respond to the series of notification messages that move the process to completion. See: Reviewing Notifications Via Electronic Mail, *Oracle Workflow User's Guide* and Viewing Notifications from a Web Browser, *Oracle Workflow User's Guide*.

   You can also access the Workflow Monitor to view the status of the workflow process. See: Using the Find Processes Web Page, *Oracle Workflow Administrator's Guide*.

To Use the Requisition Demonstration Web Page:

1. Enter the following URL in a Web browser to access the Workflow Demonstration Web page, then click on the Requisition Approval link to display the Requisition Approval Web page:

   ```
   <webagent>/wf_demo.home
   ```

   <webagent> represents the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

   Alternatively, you can enter the following URL to directly display the Requisition Approval Web page:

   ```
   <webagent>/wf_reqdemo.create_req
   ```

   > **Important:** These are both secured pages, so if you have not yet logged on as a valid workflow user in the current Web session, you will be prompted to do so before the page appears.

2. Enter a unique requisition number.

   > **Note:** The requisition number is used as the item key for the process, so it must contain only single-byte characters.

3. Specify a unique requisition description of 80 characters or less.
4. Enter a requisition amount. The amount should be a number without formatting.
5. Use the poplist fields to specify a requisition requestor and process owner. The names on these poplists are limited to the names of the roles in the demonstration data model.

6. Following the requisition input fields is the Approval Hierarchy and Spending Authority table and a description of how the Requisition demonstration process works. The Approval Hierarchy and Spending Authority table summarizes the contents of the demonstration data model.
7. Choose Submit to initiate the Requisition process and to navigate to the Requisition Created confirmation page.
8. In addition to telling you what roles you should log in as to view the process' notifications, the confirmation page also contains a HTML link to the Workflow Monitor where you can choose View Diagram to display the process diagram for the requisition you submitted in ADMIN mode. See: Workflow Monitor, *Oracle Workflow Administrator's Guide*.
9. Select the Process Timeouts HTML link to have the background engine look for any timed out notifications and execute the next activity expected to run in the case of a time out.

   Two messages appear below this link informing you of when a timeout may occur in the process.
10. Select the Create Requisition HTML link if you wish to enter and submit another requisition in the Requisition Demonstration Web page.

The Requisition Item Type

The Requisition process is associated with an item type called Requisition. Currently there are two workflow processes associated with Requisition: Requisition Approval and Notify Approver.

If you examine the property page of Requisition, you see that it has a persistence type of Temporary and persistence number of days of 0. This means that the run time data associated with any work items for this item type are eligible for purging as soon as they complete. You also see that it calls a selector function named *WF_REQDEMO.SELECTOR*. This selector function is an example PL/SQL stored procedure that returns the name of the process to run when more than one process exists for a given item type. The selector function in this example returns `REQUISITION_APPROVAL` or `'Requisition Approval'` as the process to run.

The Requisition item type also has several attributes associated with it. These attributes reference information in the demonstration application tables. The attributes are used and maintained by function activities as well as notification activities throughout the process. The following table lists the Requisition item type attributes.

Requisition Item Type Attributes

Display Name	Description	Type	Length/Format/ Lookup Type
Forward From Username	Username of the person that the requisition is forwarded from	Role	
Forward To Username	Username of the person that the requisition is forwarded to	Role	
Requestor Username	Username of the requisition preparer	Role	
Requisition Amount	Requisition amount	Number	9,999,999,999.99
Requisition Number	Unique identifier of a requisition	Text	
Monitor URL	Monitor URL	URL	
Requisition Description	Unique user identifier of a requisition	Text	80
Requisition Process Owner	Username of the requisition owner	Role	
Reminder Requisition Document	Reminder Requisition Document is generated by PL/SQL	Document	
Note	Note	Text	
Note for reminder	Note for saving the previous responder's note	Text	

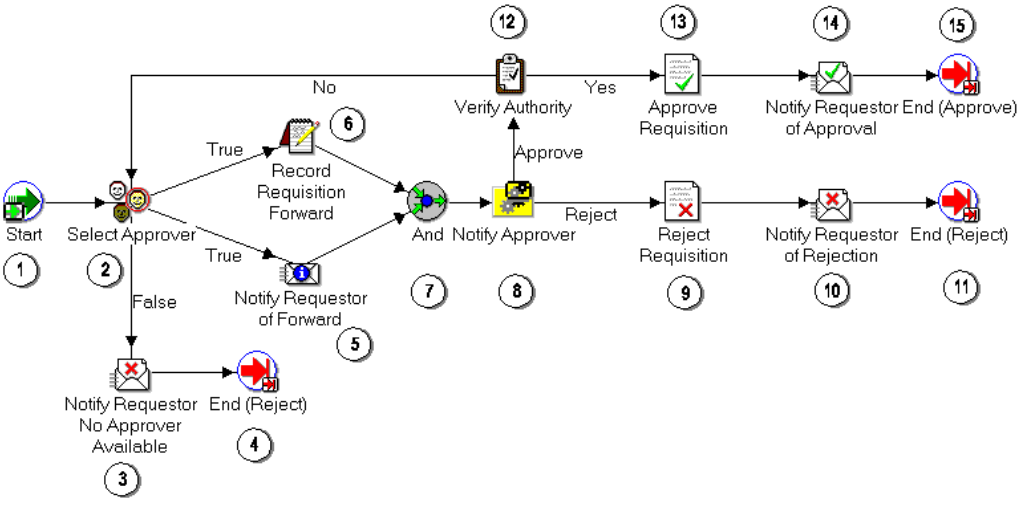
Summary of the Requisition Approval Process

To view the properties of the Requisition Approval process, select the process in the navigator tree, then choose Properties from the Edit menu. The Requisition Approval process has a result type of Approval, indicating that when the process completes, it has a result of Approve or Reject (the lookup codes in the Approval lookup type associated with the Standard item type). This process activity is also runnable, indicating that it can be initiated as a top level process to run.

The Details property page of the process activity indicates that the Requisition process has an error process assigned to it that is initiated only when an error is encountered in the process. The error process is associated with an item type called `WFERROR` and is called `DEFAULT_ERROR`. For example, if you attempt to initiate the Requisition Approval process with a requisition that is created by someone who is not listed in the employee approval hierarchy, the Workflow Engine would raise an error when it tries to execute the Select Approver activity. This error would initiate `WFERROR/DEFAULT_ERROR`, which is the Default Error Process. See: Default Error Process, page 11-6.

When you display the Process window for the Requisition Approval process, you see that the process consists of 12 unique activities, several of which are reused to comprise the 15 activity nodes that appear in the workflow diagram. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Requisition Approval Process Diagram



The Requisition workflow begins when you run a script called `wfrund.sql` or submit a requisition using the Requisition Demonstration Web page. In both cases, you must provide a requisition requestor, requisition number, requisition amount, requisition description, and process owner. See: Initiating the Requisition Workflow, page 10-6.

The workflow begins at node 1 with the Start activity.

At node 2, the process attempts to select an approver for the requisition. If an approver cannot be found for the requisition, the requestor is notified and the process ends with the final process result of Reject. If an approver is found, then the requestor is notified of who that approver is and a function records in the application that the requisition is being forwarded to the approver. Both of these activities must complete before the approver is actually notified in node 8.

Node 8 is a subprocess that requests the approver to approve the requisition by a specified period of time and if the approver does not respond by that time, the subprocess performs a timeout activity to keep sending a reminder to the approver until the approver responds. If the approver rejects the requisition, the requisition gets updated as rejected in node 9, and the requestor is notified in node 10. The process ends at this point with a result of Reject.

If the approver approves the requisition, the process transitions to node 12 to verify that the requisition amount is within the approver's spending limit. If it is, the process approves the requisition in node 13, and notifies the requestor in node 14. The process ends in this case with a result of Approve.

Requisition Process Activities

Following is a description of each activity listed by the activity's display name. You can create all the components for an activity in the graphical Oracle Workflow Builder except for the PL/SQL stored procedures that the function activities call. Function activities can execute functions external to the database by integration with Oracle Advanced Queuing or execute PL/SQL stored procedures which you must create and store in the Oracle Database. All the function activities in the Requisition process execute PL/SQL stored procedures. The naming convention for the PL/SQL stored procedures used in the Requisition process is:

`WF_REQDEMO.<PROCEDURE>`

`WF_REQDEMO` is the name of the package that groups all the procedures used by the Requisition process. *<PROCEDURE>* represents the name of the procedure.

Several activities are described in greater depth to give you an idea of how they are constructed. See: Example Function Activities, page 10-21 and Example Notification Activity, page 10-26.

Start (Node 1)

This is a Standard function activity that simply marks the start of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

Select Approver (Node 2)

This function activity determines who the next approver is for the requisition by checking the employee approval hierarchy table. This activity also saves the name of the previous approver or the name of the preparer if the requisition was never approved before. If an approver is found, this procedure returns a value of `'T'`, for True, otherwise it returns a value of `'F'` for False.

Function
WF_REQDEMO.SelectApprover

Result Type
Boolean

Prerequisite Activities
None

Notify Requestor No Approver Available (Node 3)

This activity notifies the requisition preparer that no appropriate approver could be found for the requisition. The message includes 'Send' attributes that display the requisition number, requisition description, requisition amount, and who the last approver was, if there was any. The message also includes a 'Send' attribute for the from role that sent the notification, to be displayed in the notification header.

This activity occurs in process node 3. If you display the property page of the node, you see that the activity is assigned to a performer whose name is stored in an item type attribute named Requestor Username.

Message
Requisition No Approver Found

Result Type
None

Prerequisite Activities
Select Approver

Notify Requestor of Forward (Node 5)

This activity notifies the requisition preparer that the requisition was forwarded for approval. The message includes 'Send' attributes that display the requisition number, requisition description, requisition amount, name of the approver that the requisition is forwarded to, name of the previous approver, if any, and the most recent comments appended to the requisition. The message also includes a 'Send' attribute for the from role that sent the notification, to be displayed in the notification header.

If you display the property page of this node, you see that the activity is assigned to a performer whose name is stored in an item type attribute named Requestor Username.

Message
Requisition Forward

Result Type
None

Prerequisite Activities
Select Approver

Record Requisition Forward (Node 6)

Currently this activity does nothing. It represents a point in the process where you can integrate a recording function. If you have a Purchasing/Requisition application into which you want to integrate this workflow, you can customize this activity to execute a PL/SQL stored procedure that updates your purchasing/requisition application table to indicate that the requisition is being forwarded to the next approver.

Function
WF_REQDEMO.Forward_Req

Result Type
None

Prerequisite Activities
Select Approver

And (Node 7)

This Standard function activity merges two or more parallel branches in the flow only when the activities in all of those branches complete.

Function
WF_STANDARD.ANDJOIN

Result Type
None

Prerequisite Activities
Must have at least two separate activities that each transition into this activity.

Notify Approver (Node 8)

This activity is a subprocess that notifies the approver that an action needs to be taken to either approve or reject the requisition. To view the subprocess, double-click on Notify Approver under the Processes branch in the navigator tree. The subprocess sends a notification to the approver and if the approver does not respond within a specified time, sends another reminder notification to the approver to take action. See: Summary of the Notify Approver Subprocess, page 10-15.

Result Type
Approval

Prerequisite Activities
Select Approver

Reject Requisition (Node 9)

Currently this activity does nothing. It represents a point in the process where you can integrate a function. If you have a Purchasing/Requisition application into which you want to integrate this workflow, you can customize this activity to execute a PL/SQL stored procedure that updates your purchasing/requisition application table to indicate that the requisition is rejected.

Function
WF_REQDEMO.Reject_Req

Result Type
None

Prerequisite Activities
Select Approver, Notify Approver

Notify Requestor of Rejection (Node 10)

This activity notifies the requisition preparer that the requisition was rejected. The message includes 'Send' attributes that display the requisition number, requisition description, requisition amount, name of the manager that rejected the requisition, and comments from that manager. The message also includes a 'Send' attribute for the from role that sent the notification, to be displayed in the notification header.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in an item type attribute named Requestor Username.

Message
Requisition Rejected

Result Type
None

Prerequisite Activities
Notify Approver

Verify Authority (Node 12)

This function activity verifies whether the current approver has sufficient authority to approve the requisition. The procedure compares the requisition amount with the approver's approval limit amount and returns a value of `'Y'` for Yes or `'N'` for No. If your business rules are not sensitive to the amount that an approver can approve, then you can remove this activity to customize the process.

Function
WF_REQDEMO.VerifyAuthority

Result Type
Yes/No

Prerequisite Activities
Select Approver and Notify Approver

Approve Requisition (Node 13)

Currently this activity does nothing. It represents a point in the process where you can integrate a function. If you have a Purchasing/Requisition application into which you want to integrate this workflow, you can customize this activity to execute a PL/SQL stored procedure that updates your purchasing/requisition application table to indicate that the requisition is approved.

Function
WF_REQDEMO.Approve_Req

Result Type
None

Prerequisite Activities
Select Approver, Notify Approver, Verify Authority

Notify Requestor of Approval (Node 14)

This activity notifies the requisition preparer that the requisition was approved. The message includes 'Send' attributes that display the requisition number, requisition description, requisition amount, approver name, and comments from the approver. The message also includes a 'Send' attribute for the from role that sent the notification, to be displayed in the notification header.

If you display the property page of the activity node, you see that the activity is assigned to a performer whose name is stored in an item type attribute named Requestor Username.

Message
Requisition Approved

Result Type
None

Prerequisite Activities
Select Approver, Notify Approver, Verify Authority

End (Nodes 4, 11, and 15)

This function activity marks the end of the process. Although the activity itself does not have a result type, each node of this activity in the process must have a process result assigned to it. The process result is assigned in the property page of the activity

node. Since the Requisition process activity has a result type of Approval, each End activity node must have a process result matching one of the lookup codes in the Approval lookup type.

Function
WF_STANDARD.NOOP

Result Type
None

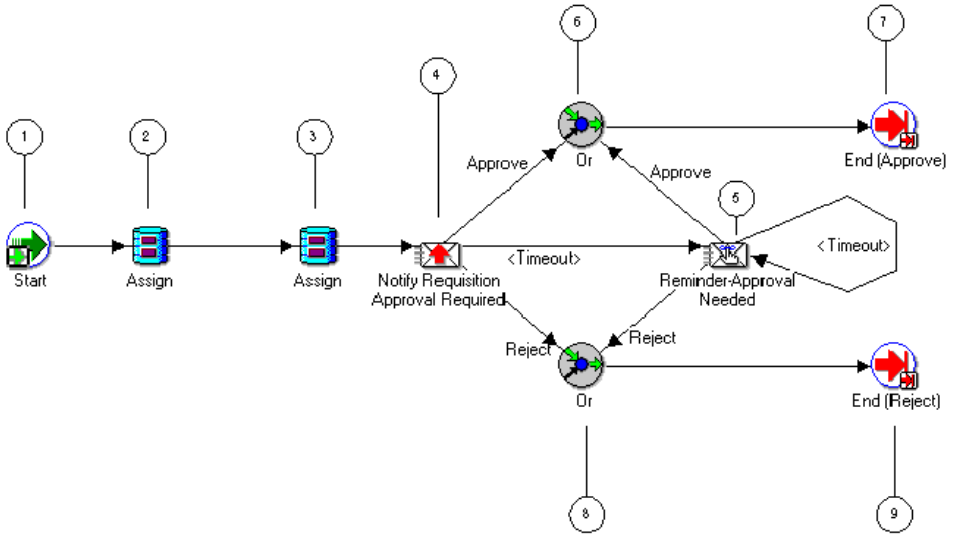
Prerequisite Activities
Start

Summary of the Notify Approver Subprocess

To view the properties of the Notify Approver subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. The Notify Approver subprocess has a result type of Approval, indicating that when the subprocess completes, it has a result of Approve or Reject (based on the lookup codes in the Approval lookup type). It is not runnable, indicating that it cannot be initiated as a top level process to run, but rather can only be run when called by another higher level process as a subprocess.

When you display the Process window for the Notify Approver subprocess, you see that the subprocess consists of 6 unique activities, several of which are reused to comprise the 9 activity nodes that appear in the workflow diagram. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Notify Approver Process Diagram



The subprocess begins at node 1 with the Start activity. At node 2, the comments from the previous approver, if any, which are stored in the Note item attribute, are assigned to the Note for Reminder item attribute for use in the Reminder-Approval Needed notification. Then the Note item attribute is cleared, so that it can be used to store comments from the current approver.

At node 4, the process notifies the current approver to approve a requisition within a specified period of time. If the approver approves the requisition, the subprocess ends at node 7 and returns the result Approve to the top level Requisition process. Similarly, if the approver rejects the requisition, the subprocess ends at node 9 and returns the result Reject to the top level Requisition process.

If the approver does not respond by the due date, the subprocess takes the `<Timeout>` transition to node 5 to send a reminder to the approver to approve the requisition. Node 5 also has a timeout value assigned to it, and if the approver does not respond to the reminder by that time, the subprocess takes the next `<Timeout>` transition to loop back to node 5 to send another reminder to the approver. This loop continues until the approver approves or rejects the requisition, which would end the subprocess at node 7 or 9, respectively.

Notify Approver Subprocess Activities

Following is a description of each activity in the Notify Approver subprocess, listed by the activity's display name.

Start (Node 1)

This is a Standard function activity that simply marks the start of the subprocess.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

Assign (Node 2)

This is a Standard function activity that assigns a value to an item attribute. This Assign activity node assigns the value of the Note item attribute to the Note for Reminder item attribute.

Function
WF_STANDARD.ASSIGN

Result Type
None

Prerequisite Activities
None

Assign (Node 3)

This is a Standard function activity that assigns a value to an item attribute. This Assign activity node assigns a blank value to the Note item attribute to clear any previous value.

Function
WF_STANDARD.ASSIGN

Result Type
None

Prerequisite Activities
None

Notify Requisition Approval Required (Node 4)

This activity notifies the approver that the requisition needs to be approved or rejected. This activity must be completed within 5 minutes, otherwise it times out.

The message includes 'Send' attributes that display the requisition number, requisition description, requisition amount, previous approver name, and preparer name for the requisition when the notification is sent. The special `WF_NOTIFICATION()` message function is called to display these attributes in a message attribute table within the message, and also to include a notification history table in the message. The message also includes a 'Send' attribute for the from role that sent the notification, to be displayed in the notification header.

The message includes a special RESULT attribute and a "Respond" attribute. The RESULT attribute has a display name of Action and prompts the approver to respond with a value of `'APPROVE'` or `'REJECT'` from the lookup type called Approval. The value that the approver selects becomes the result that determines which activity branch the Workflow Engine transitions to next.

The "Respond" attribute is called Note and this attribute prompts the approver for optional comments to include in the notification response.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in an item type attribute named Forward To Username.

Message
Requisition Approval Required

Result Type
Approval

Prerequisite Activities
Select Approver

Reminder-Approval Needed (Node 5)

This activity occurs only if the Notify Requisition Approval Required activity times out before being completed. This activity sends a reminder notice to the approver that the requisition needs to be approved or rejected.

The message includes 'Send' attributes that display the requisition number, requisition description, requisition amount, previous approver name, and preparer name for the requisition when the notification is sent. The special `WF_NOTIFICATION()` message function is called to include a notification history table in the message. The message also includes a 'Send' attribute for the from role that sent the notification, to be displayed in the notification header.

The message includes a special RESULT attribute and a "Respond" attribute. The RESULT attribute has a display name of Action and prompts the approver to respond

with a value of `'APPROVE'` or `'REJECT'` from the lookup type called Approval. The value that the approver selects becomes the result that determines which activity branch the Workflow Engine transitions to next.

The "Respond" attribute is called Note and this attribute prompts the approver for optional comments to include in the notification response.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in an item type attribute named Forward To Username.

Message
Requisition Approval Required Reminder

Result Type
Approval

Prerequisite Activities
Select Approver, Notify Requisition Approval Required

Or (Nodes 6 and 8)

This Standard function activity merges two or more parallel branches in a flow as soon as an activity in any one of those branches completes.

Function
WF_STANDARD.ORJOIN

Result Type
None

Prerequisite Activities
None

End (Nodes 7 and 9)

This function activity marks the end of the subprocess. Although the activity itself does not have a result type, each node of this activity in the subprocess must have a process result assigned to it. The process result is assigned in the property page of the activity node. Since the Notify Approver process activity has a result type of Approval, each End activity node must have a process result matching one of the lookup codes in the Approval lookup type.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
Start

Sample StartProcess Function

Both `wfrund.sql` and the Requisition Demonstration Web page call a PL/SQL stored procedure named *WF_REQDEMO.StartProcess* to initiate the Requisition process.

To examine *StartProcess* in more detail, we divide the procedure into several sections and number each section with the notation **(1)** for easy referencing. The numbers themselves are not part of the procedure.

```
(1)  procedure StartProcess
       (RequisitionNumber in varchar2,
        RequisitionDesc in varchar2,
        RequisitionAmount in number,
        RequestorUsername in varchar2,
        ProcessOwner in varchar2,
        Workflowprocess in varchar2 default null,
        item_type in varchar2 default null) is

(2)  ItemType varchar2(30) := nvl(item_type, 'WFDEMO');
     ItemKey varchar2(30) := RequisitionNumber;
     ItemUserKey varchar2(80) := RequisitionDesc;

(3)  begin
       wf_engine.CreateProcess (itemtype => ItemType,
         itemkey  => ItemKey,
         process  => WorkflowProcess );

(4)    wf_engine.SetItemUserKey (itemtype => itemtype,
         itemkey  => itemkey,
         userkey => ItemUserKey);

(5)    wf_engine.SetItemAttrText (itemtype => itemtype,
         itemkey  => itemkey,
         aname => 'REQUISITION_NUMBER',
         avalue => RequisitionNumber);

(6)    wf_engine.SetItemAttrText (itemtype => itemtype,
         itemkey  => itemkey,
         aname => 'REQUISITION_DESCRIPTION',
         avalue => ItemUserKey);

(7)    wf_engine.SetItemAttrNumber (itemtype => itemtype,
         itemkey  => itemkey,
         aname => 'REQUISITION_AMOUNT',
         avalue => RequisitionAmount);

(8)    wf_engine.SetItemAttrText (itemtype => itemtype,
         itemkey  => itemkey,
         aname => 'REQUESTOR_USERNAME',
         avalue => RequestorUsername);

(9)    wf_engine.SetItemAttrText (itemtype => itemtype,
         itemkey  => itemkey,
         aname => 'FORWARD_TO_USERNAME',
         avalue => RequestorUsername);

(10)   wf_engine.SetItemAttrText (itemtype => itemtype,
         itemkey  => itemkey,
         aname => 'REQUISITION_PROCESS_OWNER',
         avalue => ProcessOwner);

(11)   wf_engine.SetItemAttrText (itemtype => itemtype,
         itemkey  => itemkey,
         aname => 'MONITOR_URL',
         avalue => wf_monitor.GetDiagramUrl
           (WF_CORE.Translate('WF_WEB_AGENT'),
             itemtype,itemkey,'NO'));
```

```
(12)   wf_engine.SetItemAttrText (itemtype => itemtype,
         itemkey  => itemkey,
         aname => 'REM_DOCUMENT',
         avalue => 'PLSQL:wf_reqdemo.reminder_req_document/'
           ||ItemType||':'||ItemKey);

(13)   wf_engine.SetItemOwner (itemtype => itemtype,
         itemkey  => itemkey,
         owner => ProcessOwner);

(14)   wf_engine.StartProcess (itemtype => itemtype,
         itemkey => itemkey );

(15)  exception
     when others then
       wf_core.context('WF_REQDEMO','StartProcess',
         RequisitionNumber,RequisitionAmount,
         RequestorUsername,ProcessOwner,Workflowprocess);
       raise;

(16)  end StartProcess;
```

(1) This section represents the specification of the procedure, which includes the list of parameters that must be passed to *StartProcess*. It uses the same parameter values that you pass to the `wfrund.sql` script or to the field values entered in the Requisition Demonstration Web page (`WF_REQDEMO.Create_Req`).

(2) The declarative part of the procedure body begins in this section. *StartProcess* consists of calls to various Workflow Engine PL/SQL APIs. See: Workflow Engine APIs, *Oracle Workflow API Reference*.

Since all of these APIs require an item type and item key input, we define `ItemType` and `ItemKey` as local arguments. The argument `ItemType` is defined as `'WFDEMO'`, which is the internal name for the Requisition item type. The argument `ItemKey` is the value of the `RequisitionNumber` parameter that is passed to the *StartProcess* procedure.

> **Note:** The item key for a process instance can only contain single-byte characters. It cannot contain a multibyte value.

(3) The executable part of the procedure body begins here. This section calls the *CreateProcess* Workflow Engine API. This API creates a new runtime instance of the Requisition process, whose internal name is `'WFDEMO'`, and the new instance is identified by the item type and item key that are supplied. See: CreateProcess, *Oracle Workflow API Reference*.

> **Note:** If you do not pass a value for *`<process_int_name>`* when initiating the demo, the selector function for the Requisition item type determines what process to run.

(4) This section calls the SetItemUserKey Workflow Engine API to mark the new runtime instance of the Requisition process with an end-user key. The end-user key makes it easier for users to query and identify the process instance when it is displayed. See: SetItemUserKey, *Oracle Workflow API Reference*.

(5), **(6)**, **(7)**, **(8)**, **(9)**, **(10)**, **(11)**, and **(12)** These sections call either the *SetItemAttributeText* or *SetItemAttributeNumber* Workflow Engine APIs to set values for the item type attributes defined for this process. The attributes are `REQUISITION_NUMBER`, `REQUISITION_DESCRIPTION`, `REQUISITION_AMOUNT`, `REQUESTOR_USERNAME`, `FORWARD_TO_USERNAME`, `REQUISITION_PROCESS_OWNER`, `MONITOR_URL`, and `REM_DOCUMENT`. See: SetItemAttribute, *Oracle Workflow API Reference*.

(13) This section calls the SetItemOwner Workflow Engine API to mark the new runtime instance of the Requisition process with a process owner user name. Users can query for process instances by process owner. See: SetItemOwner, *Oracle Workflow API Reference*.

(14) This section calls WF_CORE.CONTEXT() if an exception occurs, to include context information in the error stack to help you locate the source of an error. See: CONTEXT, *Oracle Workflow API Reference*.

(15) This section calls the Oracle Workflow Engine *StartProcess* API to invoke the Requisition process for the item type and item key specified. See: StartProcess, *Oracle Workflow API Reference*.

(16) This section marks the end of the procedure.

Example Function Activities

In general, a function activity must have the following information specified in its Activity property page:

- Internal name for the activity.
- Display name for the activity.
- Result type for the activity, which can be none or the name of a predefined lookup type.
- Name of the PL/SQL stored procedure that the activity calls.

Also, the PL/SQL stored procedure that a function activity calls must comply with a specific API. See: Standard API for PL/SQL Procedures Called by Function Activities, page 6-2.

You can view the scripts that create the *WF_REQDEMO* stored procedure package used by the Requisition process in the `ORACLE_HOME/wf/demo` directory on your server.

Related Topics

Example: Select Approver, page 10-21

Example: Verify Authority, page 10-24

Example: Select Approver

The Select Approver function activity calls a PL/SQL stored procedure named *WF_REQDEMO.SelectApprover* that determines who the next approver is based on the employee approval hierarchy in the demonstration data model.

Result Type

This activity expects a response of `'T'` if an approver is found or `'F'` if an approver is not found. The possible responses are defined in a lookup type called Boolean, associated with the Standard item type.

PL/SQL Stored Procedure

The PL/SQL stored procedure that this function activity calls is described in detail below. Each section in the procedure is numbered with the notation **(1)** for easy referencing.

```
procedure SelectApprover
  (itemtype in varchar2,
   itemkey in varchar2,
   actid in number,
   funcmode in varchar2,
   resultout out varchar2) is

(1)  l_forward_from_username varchar2(320);
     l_forward_to_username varchar2(320);

(2)  begin
     if (funcmode = 'RUN') then
       l_forward_to_username := wf_engine.GetItemAttrText (
         itemtype => itemtype,
         itemkey => itemkey,
         aname => 'FORWARD_TO_USERNAME');

(3)    if (l_forward_to_username is null) then
         l_forward_to_username := wf_engine.GetItemAttrText (
           itemtype => itemtype,
           itemkey => itemkey,
           aname => 'REQUESTOR_USERNAME');
       end if;

(4)    l_forward_from_username := l_forward_to_username;

(5)    wf_engine.SetItemAttrText (itemtype => itemtype;
         itemkey => itemkey,
         aname => 'FORWARD_FROM_USERNAME';
         avalue => l_forward_from_username);

(6)    l_forward_to_username := wf_reqdemo.GetManager(
         l_forward_from_username);

(7)    wf_engine.SetItemAttrText (itemtype => itemtype;
         itemkey => itemkey,
         aname => 'FORWARD_TO_USERNAME';
         avalue => l_forward_to_username);

(8)    if (l_forward_to_username is null) then
         resultout :='COMPLETE:F';
       else
         resultout :='COMPLETE:T';
       end if;

(9)  end if;

(10) if (funcmode = 'CANCEL') then
       resultout :='COMPLETE';
       return;
     end if;

(11) if (funcmode = 'TIMEOUT') then
```

```
      resultout :='COMPLETE';
      return;
    end if;

(12) exception
      when others then
        wf_core.context('WF_REQDEMO','SelectorApprover',itemtype,
          itemkey,actid,funcmode);
        raise;

(13) end SelectApprover;
```

(1) The local arguments `l_forward_from_username` and `l_forward_to_username` are declared in this section.

(2) If the value of `funcmode` is `RUN`, then retrieve the name of the last person that this requisition was forwarded to for approval by assigning `l_forward_to_username` to the value of the `FORWARD_TO_USERNAME` item type attribute, determined by calling the Workflow Engine API *GetItemAttrText*. See: GetItemAttribute, *Oracle Workflow API Reference*.

(3) If the value of `l_forward_to_username` is null, then it means that the requisition has never been forwarded for approval. In this case, assign it the value of the `REQUESTOR_USERNAME` item type attribute, determined by calling the Workflow Engine API *GetItemAttrText*.

(4) Assign `l_forward_from_username` to the value of `l_forward_to_username`.

(5) This section assigns the value of `l_forward_from_username` to the `FORWARD_FROM_USERNAME` item type attribute by calling the Workflow Engine *SetItemAttrText* API.

(6) This section calls the function *GetManager* to return the manager of the previous approver stored in `l_forward_from_username`, from the WF_REQDEMO_EMP_HIERARCHY table and assigns that manager's name to `l_forward_to_username`.

(7) This section assigns the value of `l_forward_to_username` to the `FORWARD_TO_USERNAME` item type attribute by calling the Workflow Engine *SetItemAttrText* API.

(8) If `l_forward_to_username` is null, meaning there is no manager above the previous approver in the hierarchy, then assign `resultout` to be `COMPLETE:F`. Otherwise, assign `resultout` to be `COMPLETE:T`.

(9) This ends the check on `funcmode =' RUN'`.

(10) If the value of `funcmode` is `CANCEL`, then assign `resultout` to be `COMPLETE`.

(11) If the value of `funcmode` is `TIMEOUT`, then assign `resultout` to be `COMPLETE`.

(12) This section calls WF_CORE.CONTEXT if an exception occurs, to include context information in the error stack to help you locate the source of an error. See: CONTEXT, *Oracle Workflow API Reference*.

(13) This section marks the end of the procedure.

Example: Verify Authority

The Verify Authority function activity calls a PL/SQL stored procedure named *WF_REQDEMO.VerifyAuthority* to verify whether the requisition amount is within the approver's spending limit. This activity is also another example of an automated function activity that returns a result based on a business rule that you implement as a stored procedure.

Result Type

This activity expects a result of `'Yes'` or `'No'` when the procedure completes to indicate whether the approver has the authority to approve the requisition. These result values are defined in the lookup type called Yes/No, associated with the Standard item type.

PL/SQL Stored Procedure

The PL/SQL stored procedure that this function activity calls is described in detail below. Each section in the procedure is numbered with the notation **(1)** for easy referencing. We also use the convention `'l_'` to identify local arguments used within the procedure.

```
procedure VerifyAuthority
  (itemtype in varchar2,
   itemkey in varchar2,
   actid in number,
   funcmode in varchar2,
   resultout out varchar2) is

(1) l_forward_to_username varchar2(320);
     l_requisition_amount number;
     l_spending_limit number;

(2)  begin
     if (funcmode = 'RUN') then
       l_requisition_amount := wf_engine.GetItemAttrNumber (
         itemtype => itemtype,
         itemkey => itemkey,
         aname => 'REQUISITION_AMOUNT');

(3)    l_forward_to_username := wf_engine.GetItemAttrText (
         itemtype => itemtype,
         itemkey => itemkey,
         aname => 'FORWARD_TO_USERNAME');

(4)    if (wf_reqdemo.checkSpendingLimit(l_forward_to_username,
             l_requisition_amount)) then
         resultout :='COMPLETE:Y';
       else
         resultout :='COMPLETE:N';
       end if;
     end if;

(5)  if (funcmode = 'CANCEL') then
       resultout :='COMPLETE:';
       return;
     end if;

(6)  if (funcmode = 'TIMEOUT') then
       resultout :='COMPLETE:';
       return;
     end if;

(7)  exception
       when others then
         wf_core.context('WF_REQDEMO', 'VerifyAuthority',
           itemtype, itemkey, actid, funcmode);
         raise;

(8)  end VerifyAuthority;
```

(1) The local arguments `l_forward_to_username`, `l_requisition_amount`, and `l_spending_limit` are declared in this section.

(2) If the value of `funcmode` is equal to `RUN`, then assign `l_requisition_amount` to the value of the `REQUISITION_AMOUNT` item type attribute, determined by calling the Workflow Engine API *GetItemAttrNumber*. See: GetItemAttribute, *Oracle Workflow API Reference*.

(3) This section assigns `l_forward_to_username` to the value of the `FORWARD_TO_USERNAME` item type attribute, determined by calling the Workflow Engine API *GetItemAttrText*.

(4) This section calls the function *CheckSpendingLimit* for the current approver to determine whether the requisition amount is less than or equal to the approver's spending limit. If the requisition amount is less than or equal to the value in `l_spending_limit`, meaning the approver has authority to approve, then assign `resultout` to be `COMPLETE:Y`. Otherwise, assign `resultout` to be `COMPLETE:N`.

(5) If the value of `funcmode` is `CANCEL`, then assign `resultout` to be `COMPLETE:`.

(6) If the value of `funcmode` is `TIMEOUT`, then assign `resultout` to be `COMPLETE:`.

(7) This section calls WF_CORE.CONTEXT if an exception occurs, to include context information in the error stack to help you locate the source of an error. See: CONTEXT, *Oracle Workflow API Reference*.

(8) This section marks the end of the procedure.

Example Notification Activity

The Requisition process contains several notification activities that send informative messages to users. The Notify Approver subprocess, however, also includes notification activities that request a response from a user.

A notification activity requires the following information be defined in its Activity property page:

- Internal name for the activity.
- Display name for the activity.
- Result type for the activity, which can be none or the name of a predefined lookup type.
- Name of a predefined message that the notification sends out.

Related Topics

Example: Notify Requisition Approval Required, page 10-26

Example: Notify Requisition Approval Required

The Notify Requisition Approval Required activity sends a message called Requisition Approval Required to an approving manager. The message requests that the manager approve or reject a requisition and provides details about the requisition within the body of the message.

Result Type

The manager's response determines the activity that the process transitions to next. The possible responses, `'APPROVE'` or `'REJECT'`, are defined in a lookup type called Approval. These values are defined by the message's special Result attribute, whose display name is Action. These values are also the possible results of the notification activity, as defined by the Result Type field in the Activity property page.

Message

The content of the notification is defined in the message called Requisition Approval Required:

Subject

```
Requisition &REQUISITION_NUMBER, &REQUISITION_DESCRIPTION for
&REQUISITION_AMOUNT requires approval
```

Body

```
WF_NOTIFICATION(ATTRS,REQUISITION_NUMBER,REQUISITION_AMOUNT,
REQUISITION_DESCRIPTION,FORWARD_FROM_USERNAME,REQUESTOR_USERNAME)


WF_NOTIFICATION(HISTORY)
```

Message attributes, preceded by an ampersand `'&'` in the subject line or body of the message, are token substituted with runtime values when the notification is sent. However, in order for token substitution to occur properly, all message attributes referenced in the subject line or body of the message must be defined with a source of 'Send'.

In this example, the message body consists of two calls to the special message function called `WF_NOTIFICATION()`. The first call displays the requisition number, requisition description, requisition amount, previous approver name, and preparer name for the requisition in a message attribute table within the message. The second call displays a notification history table in the message. Additionally, the message includes a special 'Send' attribute named `#FROM_ROLE`, for the from role that sent the notification, to be displayed in the notification header. See: #FROM_ROLE Attribute, page 3-29 and WF_NOTIFICATION() Message Function, page 3-26.

This message also contains a special result message attribute called Action and a 'Respond' message attribute called Note.

The result message attribute is defined in the Result tab of the message's property page. The result attribute prompts the approver to respond with a value from a list of possible values provided by the lookup type specified. The response, in turn, becomes the result of the Notify Requisition Approval Required activity. In this case, the possible response values are `'APPROVE'` or `'REJECT'`, as defined by the Approval lookup type. This result determines which activity the process transitions to next.

The 'Respond' message attribute Note is of type 'Text'. This attribute prompts the approver to enter optional comments when responding to the notification.

> **Note:** To view the content of any message, double-click on the message in the navigator tree or select the message and choose Properties from the Edit menu.

Process Node Properties

If you display the properties of the Notify Requisition Approval Required activity node in the Notify Approver subprocess diagram you should see that this node is set to Normal because it is neither the start nor end activity in the process.

You should also see that the Performer is set to the Forward To Username item type attribute, indicating that the notification gets sent to the user whose name is stored in the item type attribute called 'Forward To Username'. The value of 'Forward To Username' is determined earlier in the Requisition process by the activity called Select Approver.

Event System Demonstration

The Event System Demonstration is an example of using events to transmit business documents between two systems. You initiate the demonstration process by entering a purchase order on one system. Oracle Workflow generates a purchase order XML document and sends this document to a second system. The second system processes the purchase order and sends back to the first system three XML documents representing a purchase order acknowledgement, an advanced shipment notice, and an invoice.

You can initiate this example process if you are using the standalone version of Oracle Workflow. If you are using Oracle Workflow embedded in Oracle Applications, you should consider this process mainly as an example for explanation purposes and not for demonstration use. The files necessary to set up and run this demonstration are not provided with the version of Oracle Workflow embedded in Oracle Applications.

> **Important:** For detailed information about runnable workflow processes that are integrated with Oracle Applications, refer to the appropriate Oracle Applications user's guide or online documentation. See: Predefined Workflows Embedded in Oracle E-Business Suite, page C-1.

Before running the Event System Demonstration, you must set up a Buyer system and a Supplier system to use in the demonstration.

> **Note:** You can either set up two separate systems, or you can use the same system as both the Buyer and the Supplier.

Then you can initiate an Event System Demonstration process on the Buyer system from the Workflow Demonstrations Web page. When you initiate the process, you must specify an order number, item number, item description, deliver date, total amount, and order requestor role for the purchase order.

> **Note:** The order number is used in the item key for the process, so it must contain only single-byte characters.

When you submit the purchase order, the order information is inserted into a database table and a purchase order event is raised with the order number as the event key. Raising the purchase order event triggers two subscriptions to this event that have the source type Local. The first subscription adds a correlation ID to the event message. The correlation ID consists of the prefix `PO` followed by the event key (the order number).

Because the second subscription to this event requires the complete event data, the Event Manager runs the Generate function for the event to produce a valid purchase order XML document. The second subscription sends the event to the Buyer: Top Level PO process in the Event System Demonstration item type. The Workflow Engine creates a new instance of this process with the correlation ID as the item key.

During the purchase order processing in the Buyer: Top Level PO process, a standard external Java function activity retrieves the name of the order requestor from the purchase order XML document, so that the process can send notifications to the requestor. The purchase order event message is sent to the Supplier system, and then the process waits to receive responses from the supplier.

When the purchase order event arrives at the Supplier system, two subscriptions to this event with the source type External are triggered. The first subscription changes the

correlation ID in the event message to consist of the prefix `SO` followed by the event key (the order number). The second subscription sends the event to the Supplier: Top Level Order process in the Event System Demonstration item type. The Workflow Engine creates a new instance of this process with the new correlation ID as the item key.

During the purchase order processing in the Supplier: Top Level Order process, standard external Java function activities retrieve the item number and item description from the purchase order XML document. The process sends event messages containing the following XML documents back to the Buyer system:

- purchase order acknowledgement
- advanced shipment notice
- invoice

On the Buyer system, these events trigger subscriptions with the source type External. For each event, there are two subscriptions: one that adds a correlation ID consisting of the prefix `PO` followed by the event key (the order number), and another that sends the event message to the Buyer: Top Level PO process, using the correlation ID to match the message with the running process to which it belongs. The process receives the event messages and notifies the order requestor when each one arrives. After all three response documents have been received, the process completes.

Related Topics

Installing the Event System Demonstration Data Model, page 10-30

Initiating the Event System Demonstration Workflow, page 10-31

The Event System Demonstration Item Type, page 10-34

Summary of the Buyer: Top Level PO Process, page 10-36

Buyer: Top Level PO Process Activities, page 10-37

Summary of the Buyer: Send PO to Supplier Subprocess, page 10-40

Buyer: Send PO to Supplier Subprocess Activities, page 10-40

Summary of the Buyer: Receive Supplier PO Acknowledgement Subprocess, page 10-42

Buyer: Receive Supplier PO Acknowledgement Subprocess Activities, page 10-42

Summary of the Buyer: Advanced Shipment Notice Subprocess, page 10-44

Buyer: Advanced Shipment Notice Subprocess Activities, page 10-44

Summary of the Buyer: Receive Supplier Invoicing Subprocess, page 10-46

Buyer: Receive Supplier Invoicing Subprocess Activities, page 10-46

Summary of the Supplier: Top Level Order Process, page 10-47

Supplier: Top Level Order Process Activities, page 10-48

Summary of the Supplier: Get Order Details Subprocess, page 10-51

Supplier: Get Order Details Subprocess Activities, page 10-52

Summary of the Supplier: Credit Check Subprocess, page 10-54

Supplier: Credit Check Subprocess Activities, page 10-55

Summary of the Supplier: Stock Check Subprocess, page 10-56

Supplier: Stock Check Subprocess Activities, page 10-57

Summary of the Supplier: Advanced Shipment Notice Subprocess, page 10-58

Supplier: Advanced Shipment Notice Subprocess Activities, page 10-58

Summary of the Supplier: Send Supplier Invoice Subprocess, page 10-59

Supplier: Send Supplier Invoice Subprocess Activities, page 10-60

B2B Purchase Order Event, page 10-61

B2B Purchase Order Acknowledgement Event, page 10-64

B2B Advanced Shipment Notice Event, page 10-65

B2B Invoice Event, page 10-67

Installing the Event System Demonstration Data Model

The Event System data model is installed only with the standalone version of Oracle Workflow. The data model is automatically installed for you by the Workflow Configuration Assistant. The files used in the installation are copied to the `ORACLE_HOME/demo` and `ORACLE_HOME/demo/<language>` directories of your Oracle Workflow server directory structure.

> **Important:** For the Event System Demonstration to work properly, you should perform the steps required to set up Oracle Workflow after the installation. See: Overview of Setting Up, *Oracle Workflow Administrator's Guide*.

The installation does the following:

- Calls a script called `wfevdemc.sql` to create two tables called WF_EVENTDEMO_ITEMS and WF_EVENTDEMO_PO. The table WF_EVENTDEMO_ITEMS contains the items that can be selected for the purchase order. The Event System Demonstration process updates the table WF_EVENTDEMO_PO with information from the purchase order.
- Calls the scripts `wfevdems.sql` and `wfevdemb.sql` to create the PL/SQL spec and body for a package called WF_EVENTDEMO. This package contains:
  - The PL/SQL stored procedures called by the function activities used in the Event System Demonstration workflow.
  - The PL/SQL procedures WF_EVENTDEMO.Create_Order and WF_EVENTDEMO.Track_Order called by the Oracle Workflow Web agent to generate the Web-based interface pages for the Event System Demonstration process demonstration.
- Loads the Event System Demonstration workflow definition from `wfevdeme.wft` into the database. You can view this process in the Oracle Workflow Builder.
- The data model for the Event System Demonstration process also includes the same demonstration directory service that is used by the sample Requisition process. See: Installing the Requisition Data Model, page 10-5.

> **Important:** For security reasons, the installation process automatically locks the database accounts for the users in the demonstration directory service after they are created. Before you

can begin using the accounts, you must unlock them using a script called `wfdemoul.sql`. See: Installing the Requisition Data Model, page 10-5.

Initiating the Event System Demonstration Workflow

The Event System Demonstration requires two Workflow-enabled systems, a Buyer system and a Supplier system. You can either set up two separate systems, or you can use the same system as both the Buyer and the Supplier. Before you can run the Event System Demonstration, you must set up the system or systems that you want to use. See: To Set Up the Event System Demonstration Workflow, page 10-31.

After the systems are set up, you can initiate the Event System Demonstration workflow using the Events: Buyer Workbench demonstration Web page on the Buyer system. Then use the Events: Track Order demonstration Web page on the Supplier system to continue processing the Event System Demonstration workflow. See: To Initiate the Event System Demonstration Workflow from the Buyer Workbench, page 10-31 and To Continue the Event System Demonstration Workflow on the Supplier System, page 10-33.

To Set Up the Event System Demonstration Workflow:

1. If you are using two separate installations of Oracle Workflow, designate one of the installations as the Buyer system and the other as the Supplier system.
2. If you are using two separate installations of Oracle Workflow, sign the two systems up with each other to exchange event messages. See: Signing Up Systems, page 8-45.
3. If you are using two separate installations of Oracle Workflow, on the Buyer system, locate the predefined subscription to the `demo.oracle.apps.wf.po.create` event with the source type Local and the phase 2. Edit this subscription by selecting the standard WF_IN agent on the Supplier system as the To agent. See: To Define an Event Subscription, page 8-38.
4. If you are using two separate installations of Oracle Workflow, on the Supplier system, locate the predefined subscription to the `demo.oracle.apps.wf.po.create` event with the source type External and the phase 2. Edit this subscription by selecting the standard WF_IN agent on the Buyer system as the To agent. See: To Define an Event Subscription, page 8-38.
5. Ensure that the Java Function Activity Agent is running on your systems. See: Setting Up the Java Function Activity Agent, *Oracle Workflow Administrator's Guide*.
6. Optionally schedule a background engine to run every 10 to 30 seconds on your systems. See: Setting Up Background Engines, *Oracle Workflow Administrator's Guide*.

   You can also run the background engine manually during the demonstration by choosing the Process Order link on the Events: Track Order page.

To Initiate the Event System Demonstration Workflow from the Buyer Workbench:

1. On the Buyer system, enter the following URL in a Web browser to access the Workflow Demonstration Web page:

   `<webagent>/wf_demo.home`

`<webagent>` represents the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

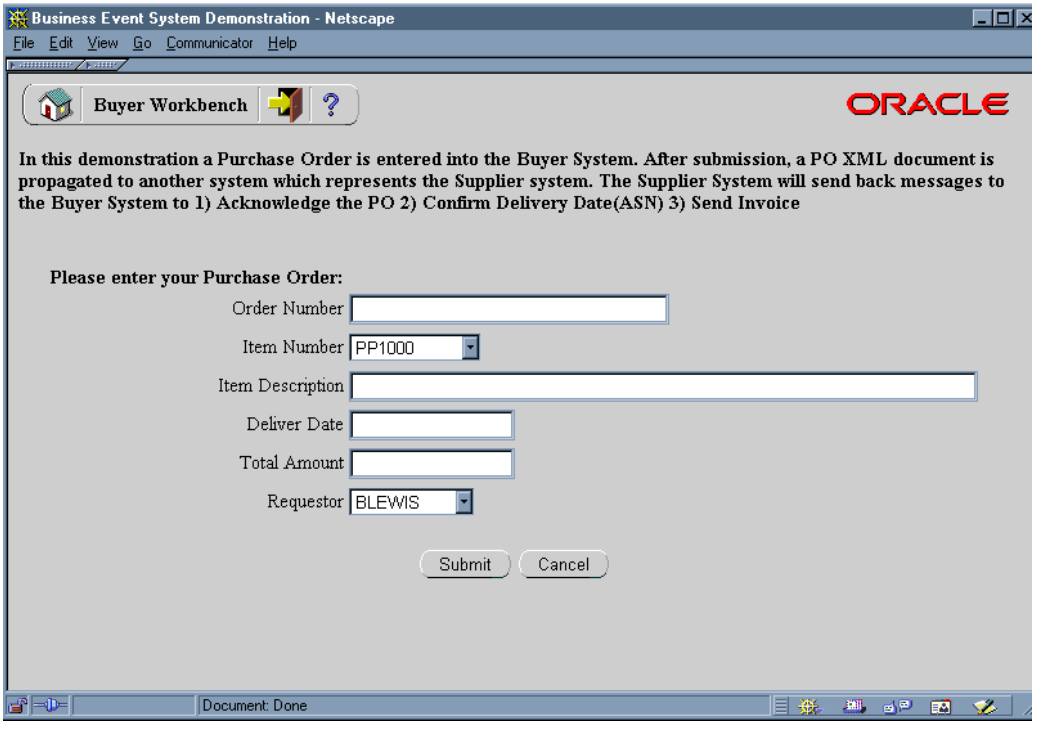
Choose the Events: Buyer Workbench link to display the Buyer Workbench Web page.

Alternatively, you can enter the following URL to directly display the Buyer Workbench Web page:

```
<webagent>/wf_eventdemo.create_order
```

> **Important:** These are both secured pages, so if you have not yet logged on as a valid workflow user in the current Web session, you will be prompted to do so before the page appears.

Buyer Workbench Page



2. Enter a unique order number for the purchase order. The order number becomes the event key for the B2B Purchase Order event that is raised.
3. Select an item number.
4. Enter an item description of 40 characters or less.
5. Enter a deliver date in the format DD-MON-YYYY.
6. Enter the total amount for the purchase order. The amount should be a number without formatting.
7. Select a role name as the order requestor.

8. Choose the Submit button to submit the purchase order and display the Events: Track Order page. You can also choose the Cancel button to return to the Workflow Demonstrations page without submitting the purchase order.
9. Choose the Workflow Monitor link to view the progress of the Buyer: Top Level PO workflow process in the Workflow Monitor. See: Workflow Monitor, *Oracle Workflow Administrator's Guide*.
10. Choose the Process Order link to run the background engine and process the deferred activity so that the buyer workflow can continue processing the purchase order.

To Continue the Event System Demonstration Workflow on the Supplier System:

1. On the Supplier system, enter the following URL in a Web browser to access the Workflow Demonstration Web page:

   `<webagent>/wf_demo.home`

   `<webagent>` represents the base URL of the Web agent configured for Oracle Workflow in your Web server. See: Setting Global User Preferences, *Oracle Workflow Administrator's Guide*.

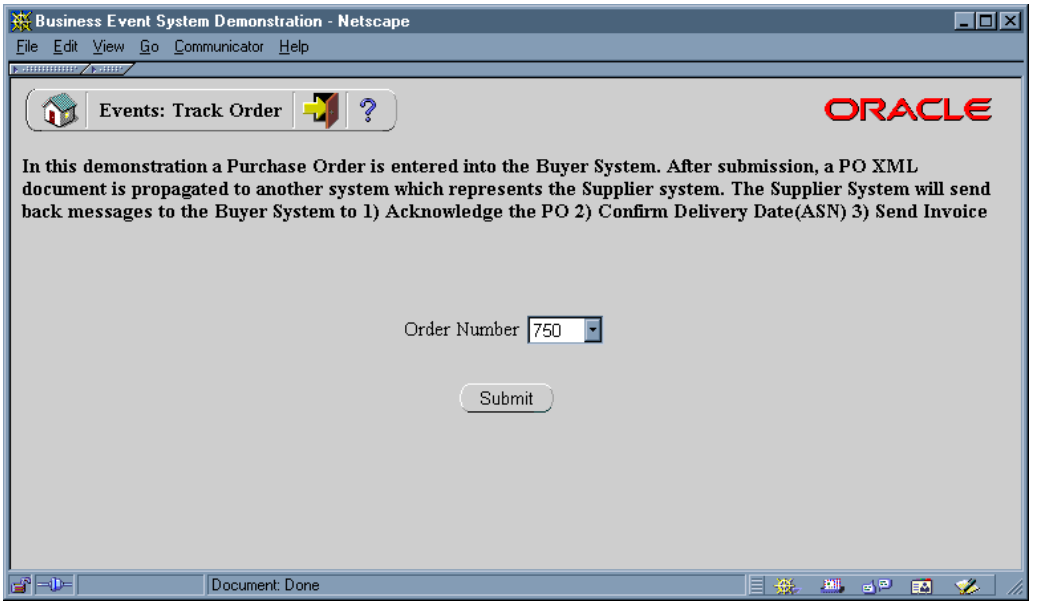
   Choose the Events: Track Order link to display the Events: Track Order Web page.

   Alternatively, you can enter the following URL to directly display the Events: Track OrderWeb page:

   `<webagent>/wf_eventdemo.track_order`

   > **Important:** These are both secured pages, so if you have not yet logged on as a valid workflow user in the current Web session, you will be prompted to do so before the page appears.

   Events: Track Order Page



2. Select the order number for your purchase order and choose the Submit button.

3. Choose the Workflow Monitor link to view the progress of the Supplier: Top Level Order process in the Workflow Monitor. See: Workflow Monitor, *Oracle Workflow Administrator's Guide*.
4. Choose the Process Order link to run the background engine and continue processing the purchase order. Perform this step twice to process both deferred activities in the supplier workflow.

The Event System Demonstration Item Type

The Event System demonstration is associated with an item type called Event System Demonstration. Currently there are eleven workflow processes associated with Event System Demonstration:

- Buyer: Top Level PO
- Buyer: Send PO to Supplier
- Buyer: Receive Supplier PO Acknowledgement
- Buyer: Advanced Shipment Notice
- Buyer: Receive Supplier Invoicing
- Supplier: Top Level Order
- Supplier: Get Order Details
- Supplier: Credit Check
- Supplier: Stock Check
- Supplier: Advanced Shipment Notice
- Supplier: Send Supplier Invoice

To view the details of the Event System Demonstration item type in the Workflow Builder, choose Open from the File menu. Then connect to the database and select the Event System Demonstration item type, or connect to a file called wfevdeme.wft in the `ORACLE_HOME\wf\Data\<language>` subdirectory on your file system.

If you examine the property page of Event System Demonstration, you see that it has a persistence type of Temporary and persistence number of days of 0. This means that the runtime data associated with any work items for this item type are eligible for purging as soon as they complete.

The Event System Demonstration item type also has several attributes associated with it. These attributes reference information in the Workflow application tables. The attributes are used and maintained by function, notification, and event activities throughout the process. The following table lists the Event System Demonstration item type attributes.

Event System Demonstration Item Type Attributes

Display Name	Description	Type	Length/Format/ Lookup Type/ Default Value
Event Name	The internal name of the original event	Text	
Event Key	The event key that uniquely identifies the specific instance of the original event	Text	
Event Message	The event message for the original event	Event	
Purchase Order Status	The purchase order status	Text	CREATED
PO Acknowledge Event	The name of the purchase order acknowledgement event	Text	
PO Acknowledge Event Key	The event key that uniquely identifies the specific instance of the purchase order acknowledgement event	Text	
PO ASN Event	The name of the advanced shipment notice event	Text	
PO ASN Event Key	The event key that uniquely identifies the specific instance of the advanced shipment notice event	Text	
PO ASN Event Message	The advanced shipment notice event message	Event	
Order Requestor	The name of the person who requested the order	Text	BLEWIS
To Agent/System 1	The inbound agent on the Buyer system that receives the event message, in the format *<agent>@<system>*	Text	
From Agent/System 1	The outbound agent on the Buyer system that sends the event message, in the format *<agent>@<system>*	Text	

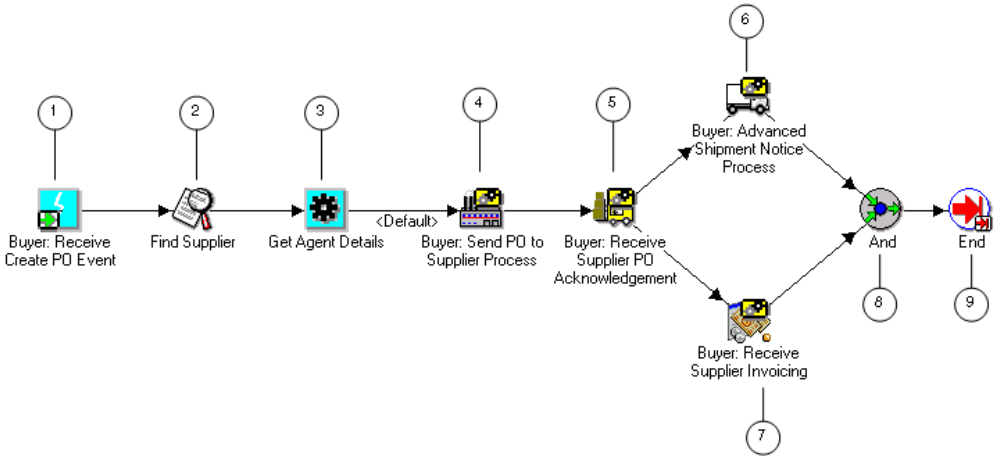
Display Name	Description	Type	Length/Format/ Lookup Type/ Default Value
To Agent/System 2	The inbound agent on the Supplier system that receives the event message, in the format *<agent>@<system>*	Text	
From Agent/System 2	The outbound agent on the Supplier system that sends the event message, in the format *<agent>@<system>*	Text	
Item Number	The item number	Text	
Item Description	The item description	Text	
Subscription GUID	The globally unique identifier of the subscription	Text	

Summary of the Buyer: Top Level PO Process

To view the properties of the Buyer: Top Level PO process, select the process in the navigator tree, then choose Properties from the Edit menu. This process activity is runnable, indicating that it can be initiated as a top level process to run.

When you display the Process window for the Buyer: Top Level PO process, you see that the process consists of nine unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Buyer: Top Level PO Process Diagram



The Buyer: Top Level PO workflow begins when you submit a purchase order from the Buyer Workbench demonstration page, raising the B2B Purchase Order event. Predefined subscriptions add a correlation ID consisting of the prefix `PO` followed

by the event key (the order number), and send the event to the Buyer: Top Level PO process. See: To Initiate the Event System Demonstration Workflow from the Buyer Workbench, page 10-31.

The process begins at node 1 with the Buyer: Receive Create PO Event activity. The process finds a supplier for the purchase order and retrieves the agent details for the intended outbound and inbound agents to send the purchase order to the supplier.

Node 4 is a subprocess that retrieves the name of the order requestor from the purchase order event message, sends the purchase order to the supplier, and notifies the order requestor that the purchase order has been sent to the supplier.

Node 5 is a subprocess that waits to receive a purchase order acknowledgement event message from the supplier. If the acknowledgement is not received within a specified period of time, the subprocess performs a timeout activity to keep notifying the order requestor that the supplier has not responded until the acknowledgement is received. When the acknowledgement is received, the subprocess notifies the order requestor of the acknowledgement.

After the purchase order acknowledgement is received, the top level process waits to receive an advanced shipment notice and an invoice from the supplier. Node 6 is a subprocess that receives the advanced shipment notice and notifies the order requestor that the order has been shipped. Node 7 is a subprocess that receives the invoice and notifies the order requestor that the invoice has arrived. After both of these subprocesses complete, the process ends.

Buyer: Top Level PO Process Activities

Following is a description of each activity in the process, listed by the activity's display name.

Buyer: Receive Create PO Event (Node 1)

This event activity receives the B2B Purchase Order event message that is sent to the Buyer: Top Level PO process by the Event Manager to start a new item.

Event Action
Receive

Event Filter
`demo.oracle.wf.b2b.po.create`

Prerequisite Activities
None

Item Attributes Set by Activity
Event Name, Event Key, Event Message

Find Supplier (Node 2)

Currently this activity does nothing. It represents a point in the process where you can integrate a function that locates a supplier for the purchase order, based on the item being ordered or other criteria.

Function
WF_EVENTDEMO.FINDSUPPLIER

Result Type
None

Prerequisite Activities
Buyer: Receive Create PO Event

Get Agent Details (Node 3)

This function activity retrieves the agent details from the subscription that sent the event message to the workflow process. The activity retrieves details for the outbound agent on the Buyer system that should send the message and the inbound agent on the Supplier system that should receive the message.

Function
WF_STANDARD.GETAGENTS

Result Type
Boolean

Prerequisite Activities
Find Supplier

Item Attributes Retrieved by Function
Subscription GUID

Item Attributes Set by Function
From Agent/System1, To Agent/System2

Buyer: Send PO to Supplier Process (Node 4)

This activity is a subprocess that retrieves the name of the order requestor from the purchase order event message, sends the purchase order to the supplier, and notifies the order requestor that the purchase order has been sent to the supplier. To view the subprocess, double-click on Buyer: Send PO to Supplier under the Processes branch in the navigator tree. See: Summary of the Buyer: Send to PO Subprocess, page 10-40.

Result Type
None

Prerequisite Activities
Get Agent Details

Buyer: Receive Supplier PO Acknowledgement (Node 5)

This activity is a subprocess that waits to receive a purchase order acknowledgement event message from the supplier. If the acknowledgement is not received within a specified period of time, the subprocess performs a timeout activity to keep notifying the order requestor that the supplier has not responded until the acknowledgement is received. When the acknowledgement is received, the subprocess notifies the order requestor of the acknowledgement.

To view the subprocess, double-click on Buyer: Receive Supplier PO Acknowledgement under the Processes branch in the navigator tree. See: Summary of the Buyer: Receive Supplier PO Acknowledgement Subprocess, page 10-42.

Result Type
None

Prerequisite Activities
Buyer: Send PO to Supplier

Buyer: Advanced Shipment Notice Process (Node 6)

This activity is a subprocess that receives an advanced shipment notice from the supplier and notifies the order requestor that the order has been shipped. To view the subprocess, double-click on Buyer: Advanced Shipment Notice under the Processes branch in the navigator tree. See: Summary of the Buyer: Advanced Shipment Notice Subprocess, page 10-44.

Result Type
None

Prerequisite Activities
Buyer:Receive Supplier PO Acknowledgement

Buyer: Receive Supplier Invoicing (Node 7)

This activity is a subprocess that receives an invoice from the supplier and notifies the order requestor that the invoice has arrived. To view the subprocess, double-click on Buyer: Receive Supplier Invoicing under the Processes branch in the navigator tree. See: Summary of the Buyer: Receive Supplier Invoicing Subprocess, page 10-46.

Result Type
None

Prerequisite Activities
Buyer:Receive Supplier PO Acknowledgement

And (Node 8)

This Standard function activity merges two or more parallel branches in the flow only when the activities in all of those branches complete.

Function
WF_STANDARD.ANDJOIN

Result Type
None

Prerequisite Activities
Buyer: Advanced Shipment Notice, Buyer: Receive Supplier Invoicing

End (Node 9)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

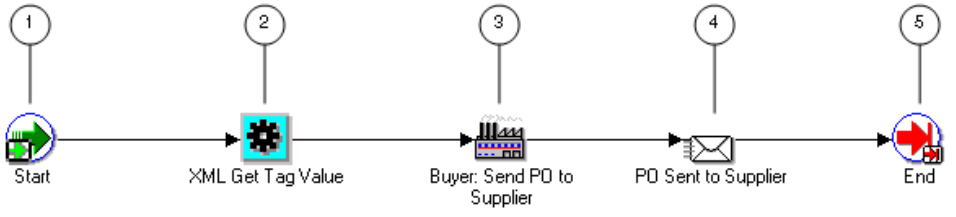
Prerequisite Activities
And

Summary of the Buyer: Send PO to Supplier Subprocess

To view the properties of the Buyer: Send PO to Supplier subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Buyer: Send PO to Supplier subprocess, you see that the subprocess consists of five unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Buyer: Send PO to Supplier Process Diagram



The subprocess begins at node 1 with the Start activity. At node 2, the process retrieves the name of the order requestor from the purchase order event message. Then the process sends the purchase order to the supplier and notifies the order requestor that the purchase order has been sent. The subprocess ends at this point.

Buyer: Send PO to Supplier Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

Start (Node 1)

This Standard function activity marks the start of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

XML Get Tag Value (Node 2)

This Standard external Java function activity retrieves the value of a particular XML tag set in the event message. At this node, the process retrieves the name of the order requestor from the purchase order event message.

Function
oracle.apps.fnd.wf.XMLGetTagValue

Result Type
None

Prerequisite Activities
Start

Item Attributes Retrieved by Function
Event Message

Item Attributes Set by Function
Order Requestor

Buyer: Send PO to Supplier (Node 3)

This event activity sends the purchase order event message to the supplier, using the outbound agent on the Buyer system and the inbound agent on the Supplier system identified by the Get Agent Details activity in the Buyer: Top Level PO process.

Event Action
Send

Prerequisite Activities
XML Get Tag Value

Item Attributes Retrieved by Activity
Event Message, From Agent/System1, To Agent/System2

PO Sent to Supplier (Node 4)

This activity notifies the order requestor that the purchase order has been sent to the supplier. The message includes a 'Send' attribute that displays the purchase order number when the notification is sent.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in the item type attribute named Order Requestor.

Message
PO Sent to Supplier

Result Type
None

Prerequisite Activities
Buyer: Send PO to Supplier

End (Node 5)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

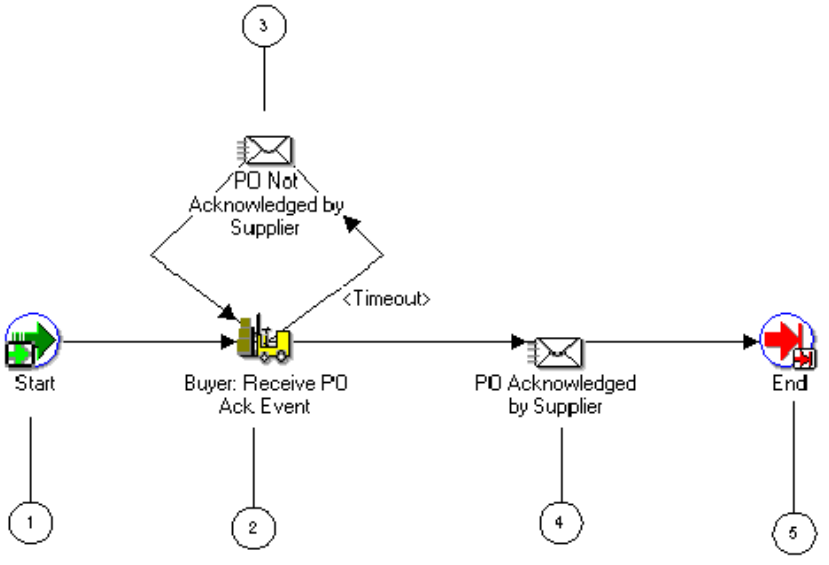
Prerequisite Activities
PO Sent to Supplier

Summary of the Buyer: Receive Supplier PO Acknowledgement Subprocess

To view the properties of the Buyer: Receive Supplier PO Acknowledgement subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Buyer: Receive Supplier PO Acknowledgement subprocess, you see that the subprocess consists of five unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Buyer: Receive Supplier PO Acknowledgement Process Diagram



The subprocess begins at node 1 with the Start activity. At node 2, the process waits to receive a purchase order acknowledgement event message from the supplier. If the acknowledgement is not received within a specified period of time, the subprocess performs a timeout activity to keep notifying the order requestor that the supplier has not responded until the acknowledgement is received. When the acknowledgement is received, the subprocess notifies the order requestor of the acknowledgement. The subprocess ends at this point.

Buyer: Receive Supplier PO Acknowledgement Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

Start (Node 1)

This Standard function activity marks the start of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

Buyer: Receive PO Ack Event (Node 2)

This event activity waits to receive the B2B Purchase Order Acknowledgement event message from the supplier. The activity must be completed within one day; otherwise it times out.

Event Action
Receive

Event Filter
`demo.oracle.wf.b2b.po.ack`

Prerequisite Activities
Start

Item Attributes Set by Activity
Event Name, Event Key, Event Message

PO Not Acknowledged by Supplier (Node 3)

This activity occurs only if the Buyer: Receive PO Ack Event activity times out before being completed. This activity notifies the order requestor that the supplier has not yet acknowledged the purchase order. The message includes a 'Send' attribute that displays the purchase order number when the notification is sent.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in the item type attribute named Order Requestor.

Message
PO Not Acknowledged

Result Type
None

Prerequisite Activities
Buyer: Receive PO Ack Event

PO Acknowledged by Supplier (Node 4)

This activity notifies the order requestor that the purchase order has been acknowledged by the supplier. The message includes a 'Send' attribute that displays the purchase order number when the notification is sent.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in the item type attribute named Order Requestor.

Message
PO Acknowledged

Result Type
None

Prerequisite Activities
Buyer: Receive PO Ack Event

End (Node 5)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

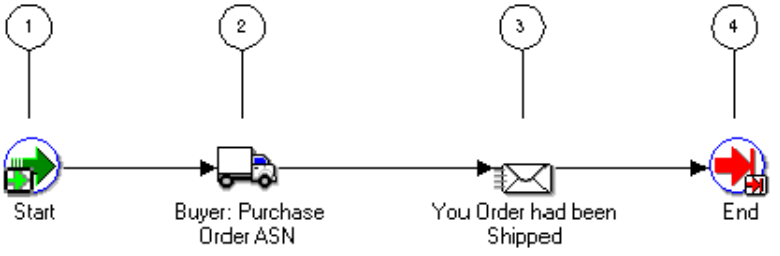
Prerequisite Activities
PO Acknowledged by Supplier

Summary of the Buyer: Advanced Shipment Notice Subprocess

To view the properties of the Buyer: Advanced Shipment Notice subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Buyer: Advanced Shipment Notice subprocess, you see that the subprocess consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Buyer: Advanced Shipment Notice Process Diagram



The subprocess begins at node 1 with the Start activity. At node 2, the process waits to receive an advanced shipment notice from the supplier. When the advanced shipment notice is received, the process notifies the order requestor that the order has been shipped. The subprocess ends at this point.

Buyer: Advanced Shipment Notice Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

Start (Node 1)

This Standard function activity marks the start of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

Buyer: Purchase Order ASN (Node 2)

This event activity waits to receive the B2B Advanced Shipment Notice event message from the supplier.

Event Action
Receive

Event Filter
`demo.oracle.wf.b2b.po.asn`

Prerequisite Activities
Start

Item Attributes Set by Activity
Event Name, Event Key, Event Message

Your Order has been Shipped (Node 3)

This activity notifies the order requestor that the purchase order advanced shipment notice has been received, indicating that the order has been shipped by the supplier. The message includes a 'Send' attribute that displays the purchase order number when the notification is sent.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in the item type attribute named Order Requestor.

Message
PO Advanced Shipment Notice

Result Type
None

Prerequisite Activities
Buyer: Purchase Order ASN

End (Node 4)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

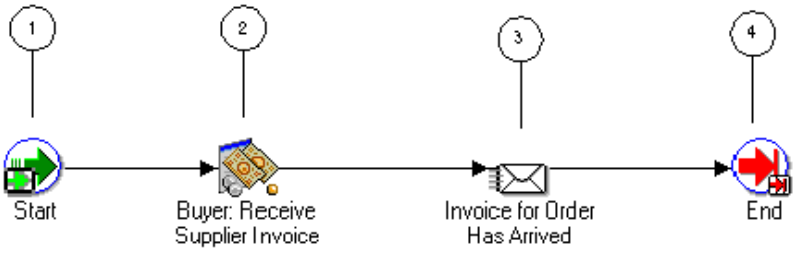
Prerequisite Activities
Your Order has been Shipped

Summary of the Buyer: Receive Supplier Invoicing Subprocess

To view the properties of the Buyer: Receive Supplier Invoicing subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Buyer: Receive Supplier Invoicing subprocess, you see that the subprocess consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Buyer: Receive Supplier Invoicing Process Diagram



The subprocess begins at node 1 with the Start activity. At node 2, the process waits to receive an invoice from the supplier. When the invoice is received, the process notifies the order requestor that the invoice has arrived. The subprocess ends at this point.

Buyer: Receive Supplier Invoicing Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

Start (Node 1)

This Standard function activity marks the start of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

Buyer: Receive Supplier Invoice (Node 2)

This event activity waits to receive the B2B Invoice event message from the supplier.

Event Action
Receive

Event Filter
`demo.oracle.wf.b2b.po.invoice`

Prerequisite Activities
Start

Item Attributes Set by Activity
Event Name, Event Key, Event Message

Invoice for Order Has Arrived (Node 3)

This activity notifies the order requestor that the invoice for the purchase order has been received from the supplier. The message includes a 'Send' attribute that displays the purchase order number when the notification is sent.

If you display the property page of this activity node, you see that the activity is assigned to a performer whose name is stored in the item type attribute named Order Requestor.

Message
Supplier PO Invoice

Result Type
None

Prerequisite Activities
Buyer: Receive Supplier Invoice

End (Node 4)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

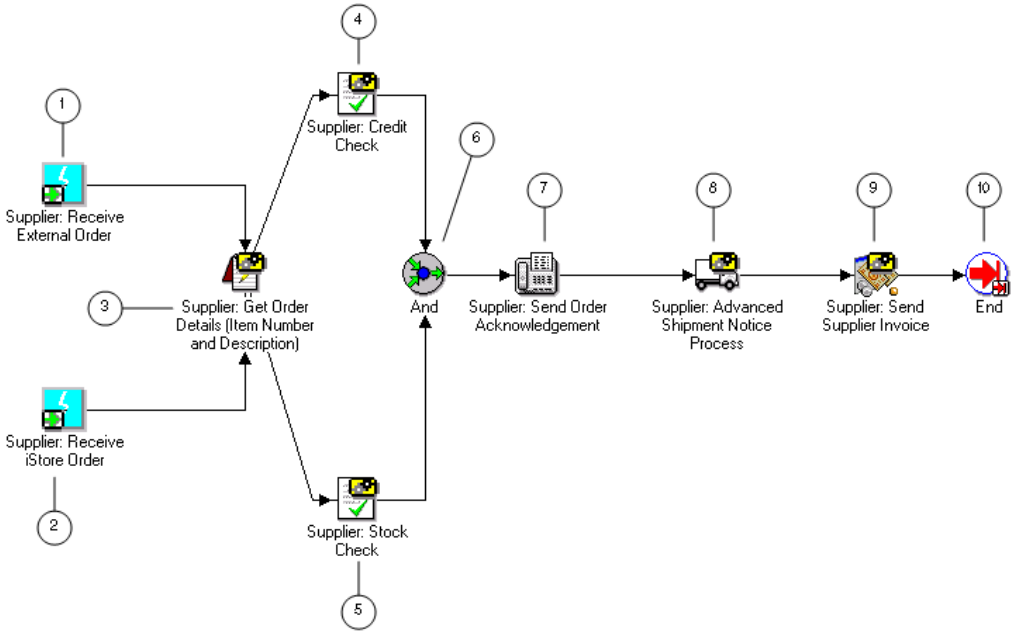
Prerequisite Activities
Invoice for Order Has Arrived

Summary of the Supplier: Top Level Order Process

To view the properties of the Supplier: Top Level Order process, select the process in the navigator tree, then choose Properties from the Edit menu. This process activity is runnable, indicating that it can be initiated as a top level process to run.

When you display the Process window for the Supplier: Top Level Order process, you see that the process consists of ten unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Supplier: Top Level Order Process Diagram



The Supplier: Top Level Order workflow begins when the Supplier system receives the B2B Purchase Order event from the Buyer system. Predefined subscriptions change the correlation ID to consist of the prefix `SO` followed by the event key (the order number), and send the event to the Supplier: Top Level Order process.

The process begins by receiving a purchase order either at node 1 or at node 2, depending on the source of the purchase order.

Node 3 is a subprocess that retrieves the agent details for the intended outbound and inbound agents to send responses to the buyer, as well as retrieving the item number and item description.

Node 4 is a subprocess that performs a credit check if the total cost of the order is greater than 2000. Node 5 is a subprocess that performs a stock check to determine if the ordered item is in stock. When both the credit check and the stock check are complete, the process sends an acknowledgement of the purchase order to the buyer.

Node 8 is a subprocess that checks the shipping details for the order and then sends an advanced shipment notice to the buyer. Node 9 is a subprocess that creates an invoice and sends the invoice to the buyer. After sending the invoice, the process ends.

Supplier: Top Level Order Process Activities

Following is a description of each activity in the process, listed by the activity's display name.

Supplier: Receive External Order (Node 1)

This event activity receives the B2B Purchase Order event message that is sent to the Supplier: Top Level Order process by the Event Manager when the event is received from the buyer.

Event Action
Receive

Event Filter
`demo.oracle.wf.b2b.po.create`

Prerequisite Activities
None

Item Attributes Set by Activity
Event Name, Event Key, Event Message

Supplier: Receive iStore Order (Node 2)

This event activity receives a purchase order event message that is sent when an order is placed through the supplier's iStore application. Currently this event is not defined. The activity represents a point in the process where you can integrate the reception of purchase orders from another source.

Event Action
Receive

Event Filter
`demo.oracle.wf.istore.po.create`

Prerequisite Activities
None

Item Attributes Set by Activity
Event Name, Event Key, Event Message

Supplier: Get Order Details (Node 3)

This activity is a subprocess that retrieves the agent details for the intended outbound and inbound agents to send responses to the buyer, as well as retrieving the item number and item description of the ordered item.

To view the subprocess, double-click on Supplier: Get Order Details under the Processes branch in the navigator tree. See: Summary of the Supplier: Get Order Details Subprocess, page 10-51.

Result Type
None

Prerequisite Activities
Supplier: Receive External Order or Supplier: Receive iStore Order

Supplier: Credit Check (Node 4)

This activity is a subprocess that performs a credit check if the total cost of the order is greater than $2000.

To view the subprocess, double-click on Supplier: Credit Check under the Processes branch in the navigator tree. See: Summary of the Supplier: Credit Check Subprocess, page 10-54.

Result Type
None

Prerequisite Activities
Supplier: Get Order Details

Supplier: Stock Check (Node 5)

This activity is a subprocess that performs a stock check to determine if the ordered item is in stock.

To view the subprocess, double-click on Supplier: Stock Check under the Processes branch in the navigator tree. See: Summary of the Supplier: Stock Check Subprocess, page 10-56.

Result Type
None

Prerequisite Activities
Supplier: Get Order Details

And (Node 6)

This Standard function activity merges two or more parallel branches in the flow only when the activities in all of those branches complete.

Function
WF_STANDARD.ANDJOIN

Result Type
None

Prerequisite Activities
Supplier: Credit Check, Supplier: Stock Check

Supplier: Send Order Acknowledgement (Node 7)

This event activity sends the B2B Purchase Order Acknowledgement event message to the buyer, using the outbound agent on the Supplier system and the inbound agent on the Buyer system identified by the Get Agent Details activity in the Supplier: Get Order Details subprocess.

Event Action
Send

Prerequisite Activities
And

Item Attributes Retrieved by Activity
Event Message, From Agent/System2, To Agent/System1

Supplier: Advanced Shipment Notice (Node 8)

This activity is a subprocess that checks the shipping details for the order and then sends an advanced shipment notice to the buyer.

To view the subprocess, double-click on Supplier: Advanced Shipment Notice under the Processes branch in the navigator tree. See: Summary of the Supplier: Advanced Shipment Notice Subprocess, page 10-58.

Result Type
None

Prerequisite Activities
Supplier: Send Order Acknowledgement

Supplier: Send Supplier Invoice (Node 9)

This activity is a subprocess that creates an invoice for the purchase order and sends the invoice to the buyer.

To view the subprocess, double-click on Supplier: Send Supplier Invoice under the Processes branch in the navigator tree. See: Summary of the Supplier: Send Supplier Invoice Subprocess, page 10-59.

Result Type
None

Prerequisite Activities
Supplier: Advanced Shipment Notice

End (Node 10)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

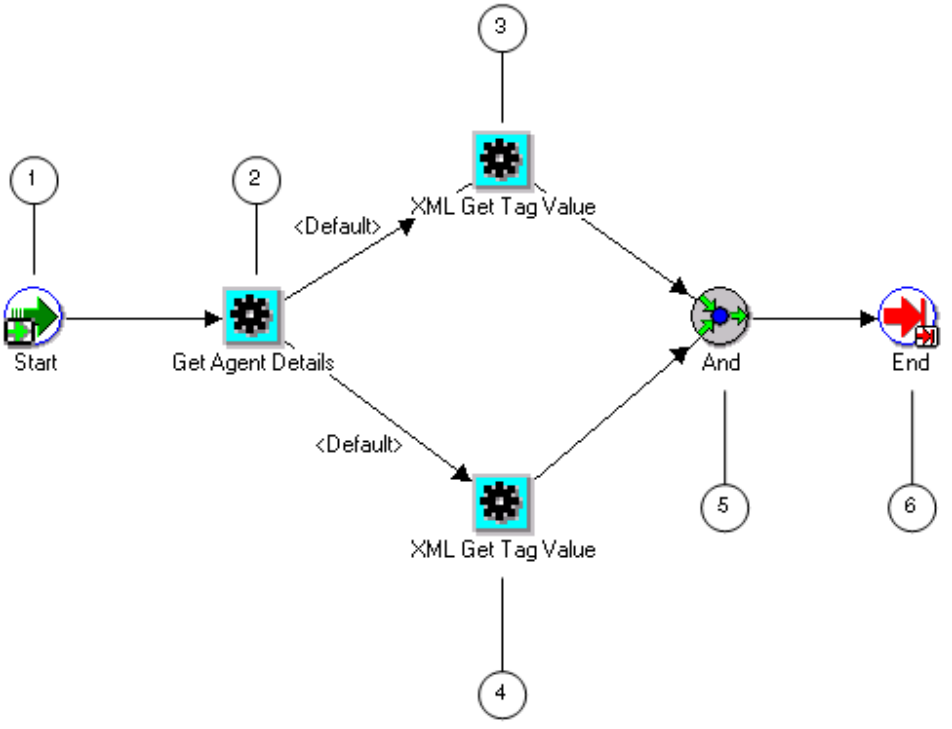
Prerequisite Activities
Supplier: Send Supplier Invoice

Summary of the Supplier: Get Order Details Subprocess

To view the properties of the Supplier: Get Order Details subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Supplier: Get Order Details subprocess, you see that the subprocess consists of five unique activities, one of which is reused to make up the six activity nodes that appear in the workflow diagram. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Supplier: Get Order Details Process Diagram



The subprocess begins at node 1 with the Start activity. At node 2, the process retrieves the agent details for the intended outbound and inbound agents to send responses to the buyer. Next, the process retrieves the item number and item description of the ordered item from the purchase order. When both the item number and item description have been retrieved, the subprocess ends.

Supplier: Get Order Details Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

Start (Node 1)

This Standard function activity marks the start of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

Get Agent Details (Node 2)

This function activity retrieves the agent details from the subscription that sent the event message to the workflow process. The activity retrieves details for the outbound agent on the Supplier system that should send the response messages and the inbound agent on the Buyer system that should receive the response messages.

Function
WF_STANDARD.GETAGENTS

Result Type
Boolean

Prerequisite Activities
Start

Item Attributes Retrieved by Function
Subscription GUID

Item Attributes Set by Function
From Agent/System2, To Agent/System1

XML Get Tag Value (Node 3)

This Standard external Java function activity retrieves the value of a particular XML tag set in the event message. At this node, the process retrieves the item number from the purchase order event message.

Function
oracle.apps.fnd.wf.XMLGetTagValue

Result Type
None

Prerequisite Activities
Get Agent Details

Item Attributes Retrieved by Function
Event Message

Item Attributes Set by Function
Item Number

XML Get Tag Value (Node 4)

This Standard external Java function activity retrieves the value of a particular XML tag set in the event message. At this node, the process retrieves the item description from the purchase order event message.

Function
oracle.apps.fnd.wf.XMLGetTagValue

Result Type
None

Prerequisite Activities
Get Agent Details

Item Attributes Retrieved by Function
Event Message

Item Attributes Set by Function
Item Description

And (Node 5)

This Standard function activity merges two or more parallel branches in the flow only when the activities in all of those branches complete.

Function
WF_STANDARD.ANDJOIN

Result Type
None

Prerequisite Activities
XML Get Tag Value (node 3), XML Get Tag Value (node 4)

End (Node 6)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

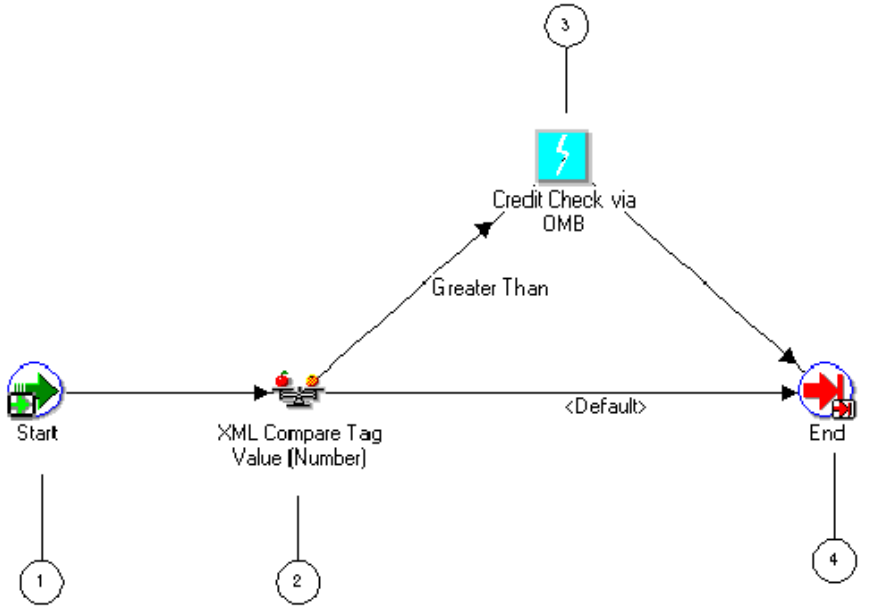
Prerequisite Activities
And

Summary of the Supplier: Credit Check Subprocess

To view the properties of the Supplier: Credit Check subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Supplier: Credit Check subprocess, you see that the subprocess consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Supplier: Credit Check Process Diagram



The subprocess begins at node 1 with the Start activity. At node 2, the process checks whether the total cost of the order is greater than 2000. If the total cost is equal to or less than 2000, the process ends at this point. If the total cost is greater than 2000, the process initiates a credit check on the buyer and then ends.

Supplier: Credit Check Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

Start (Node 1)

This Standard function activity marks the start of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

XML Compare Tag Value (Number) (Node 2)

This Standard external Java function activity compares the value of a particular XML tag set in the event message with a test value. At this node, the process compares the total cost from the purchase order event message with the test value 2000.

Function
oracle.apps.fnd.wf.XMLCompareTag

Result Type
Comparison

Prerequisite Activities
Start

Item Attributes Retrieved by Function
Event Message

Credit Check (Node 3)

This activity occurs only if the total cost of the purchase order is greater than 2000. This event activity raises a credit check event. Currently this event is not defined. The activity represents a point in the process where you can integrate a credit check.

Event Action
Raise

Prerequisite Activities
XML Compare Tag Value (Number)

Item Attributes Retrieved by Activity
Event Key

End (Node 4)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

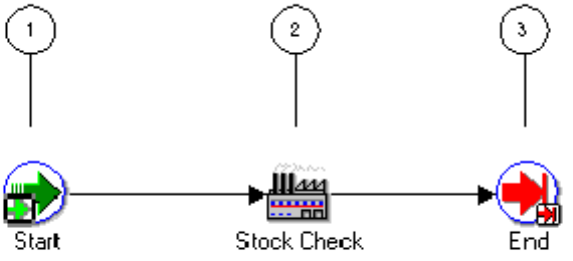
Prerequisite Activities
XML Compare Tag Value (Number)

Summary of the Supplier: Stock Check Subprocess

To view the properties of the Supplier: Stock Check subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Supplier: Stock Check subprocess, you see that the subprocess consists of three unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Supplier: Stock Check Process Diagram



The subprocess begins at node 1 with the Start activity. At node 2, the process performs a stock check to determine if the ordered item is in stock. The subprocess ends at this point.

Supplier: Stock Check Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

Start (Node 1)

This Standard function activity marks the start of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

Stock Check (Node 2)

Currently, this activity does nothing. It represents a point in the process where you can integrate a function that performs a stock check.

Function
WF_EVENTDEMO.STOCKCHECK

Result Type
None

Prerequisite Activities
Start

End (Node 3)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

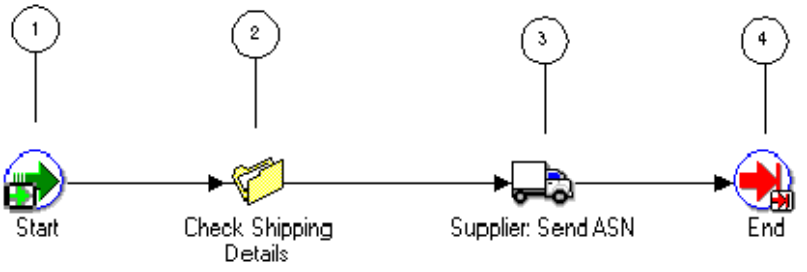
Prerequisite Activities
Stock Check

Summary of the Supplier: Advanced Shipment Notice Subprocess

To view the properties of the Supplier: Advanced Shipment Notice subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Supplier: Advanced Shipment Notice subprocess, you see that the subprocess consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Supplier: Advanced Shipment Notice Process Diagram



The subprocess begins at node 1 with the Start activity. At node 2, the process checks the shipping details for the purchase order. Then the process sends an advanced shipment notice to the buyer. The subprocess ends at this point.

Supplier: Advanced Shipment Notice Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

Start (Node 1)

This Standard function activity marks the start of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

Check Shipping Details (Node 2)

Currently, this activity does nothing. It represents a point in the process where you can integrate a function that checks the shipping details for the purchase order.

Function
WF_EVENTDEMO.CHECKSHIPPING

Result Type
None

Prerequisite Activities
Start

Supplier: Send ASN (Node 3)

This event activity sends the B2B Advanced Shipment Notice event message to the buyer, using the outbound agent on the Supplier system and the inbound agent on the Buyer system identified by the Get Agent Details activity in the Supplier: Get Order Details subprocess.

Event Action
Send

Prerequisite Activities
Check Shipping Details

Item Attributes Retrieved by Activity
Event Message, From Agent/System2, To Agent/System1

End (Node 4)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

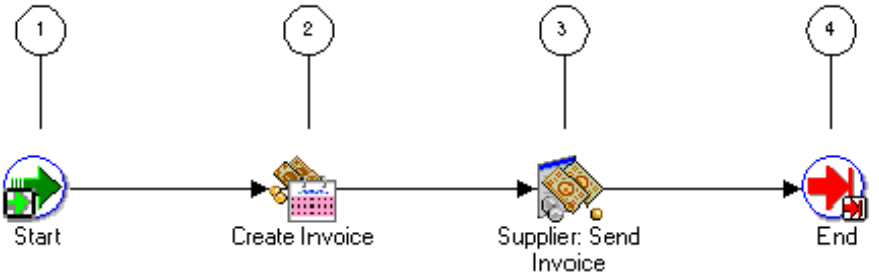
Prerequisite Activities
Supplier: Send ASN

Summary of the Supplier: Send Supplier Invoice Subprocess

To view the properties of the Supplier: Send Supplier Invoice subprocess, select its process activity in the navigator tree, then choose Properties from the Edit menu. This process activity is not runnable, indicating that it cannot be initiated as a top level process to run but must be called from a higher level process.

When you display the Process window for the Supplier: Send Supplier Invoice subprocess, you see that the subprocess consists of four unique activities. To examine the activities of the process in more detail, we have numbered each node for easy referencing below. The numbers themselves are not part of the process diagram.

Supplier: Send Supplier Invoice Process Diagram



The subprocess begins at node 1 with the Start activity. At node 2, the process creates an invoice for the items ordered in the purchase order. Then the process sends the invoice to the buyer. The subprocess ends at this point.

Supplier: Send Supplier Invoice Subprocess Activities

Following is a description of each activity in the subprocess, listed by the activity's display name.

Start (Node 1)

This Standard function activity marks the start of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
None

Create Invoice (Node 2)

Currently, this activity does nothing. It represents a point in the process where you can integrate a function that creates an invoice for the items ordered in the purchase order.

Function
WF_EVENTDEMO.CREATEINVOICE

Result Type
None

Prerequisite Activities
Start

Supplier: Send Invoice (Node 3)

This event activity sends the B2B Invoice event message to the buyer, using the outbound agent on the Supplier system and the inbound agent on the Buyer system identified by the Get Agent Details activity in the Supplier: Get Order Details subprocess.

Event Action
Send

Prerequisite Activities
Create Invoice

Item Attributes Retrieved by Activity
Event Message, Event Key, From Agent/System2, To Agent/System1

End (Node 4)

This Standard function activity marks the end of the process.

Function
WF_STANDARD.NOOP

Result Type
None

Prerequisite Activities
Supplier: Send Invoice

B2B Purchase Order Event

This event is raised on the Buyer system when you submit a purchase order from the Buyer Workbench demonstration page.

Internal Name
`demo.oracle.wf.b2b.po.create`

Status
Enabled

Generate Function
wf_eventdemo.generatexml

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides four default subscriptions to the B2B Purchase Order event. The first subscription adds a correlation ID to the event message when the B2B Purchase Order event is raised locally. The correlation ID consists of the prefix `PO` followed by the event key (the order number). This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	demo.oracle.wf.b2b.po.create
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	wf_eventdemo.derivecorrelationid
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The second subscription sends the event message to the Buyer: Top Level PO process in the Event System Demonstration item type when the B2B Purchase Order event is raised locally. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	Local
Event Filter	demo.oracle.wf.b2b.po.create
Phase	2
Status	Enabled
Rule Data	Message
Rule Function	wf_rule.workflow_protocol
Workflow Item Type	WFEVDEME
Workflow Process Name	RCVPOCRT
Out Agent	WF_OUT@*<local system>*
To Agent	WF_IN@*<local system>*
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The third subscription modifies the correlation ID in the event message when the B2B Purchase Order event is received from an external source. The correlation ID consists of the prefix `SO` followed by the event key (the order number). This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.create
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	wf_eventdemo.derivecorrelationid
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The fourth subscription sends the event message to the Supplier: Top Level Order process in the Event System Demonstration item type when the B2B Purchase Order event is received from an external source. This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.create
Phase	2
Status	Enabled
Rule Data	Key
Rule Function	wf_rule.workflow_protocol
Workflow Item Type	WFEVDEME
Workflow Process Name	SUPPORCV
Out Agent	WF_OUT@*<local system>*
To Agent	WF_IN@*<local system>*
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

B2B Purchase Order Acknowledgement Event

This event is sent from the Supplier system to the Buyer system as a purchase order acknowledgement after the B2B Purchase Order event is received and processed.

Internal Name
`demo.oracle.wf.b2b.po.ack`

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides two default subscriptions to the B2B Purchase Order Acknowledgement event. The first subscription adds a correlation ID to the event message when the B2B Purchase Order Acknowledgement event is received from an external source. The correlation ID consists of the prefix `PO` followed by the event key (the order number). This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.ack
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	wf_eventdemo.derivecorrelationid
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The second subscription sends the event message to the Buyer: Top Level PO process in the Event System Demonstration item type when the B2B Purchase Order Acknowledgement event is received from an external source. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.ack
Phase	2
Status	Enabled
Rule Data	Key
Workflow Item Type	WFEVDEME
Workflow Process Name	RCVPOCRT
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

B2B Advanced Shipment Notice Event

This event is sent from the Supplier system to the Buyer system as an advanced shipment notice after the B2B Purchase Order event is received and processed.

Internal Name
`demo.oracle.wf.b2b.po.asn`

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides two default subscriptions to the B2B Advanced Shipment Notice event. The first subscription adds a correlation ID to the event message when the B2B Advanced Shipment Notice event is received from an external source. The correlation ID consists of the prefix `PO` followed by the event key (the order number). This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.asn
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	wf_eventdemo.derivecorrelationid
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The second subscription sends the event message to the Buyer: Top Level PO process in the Event System Demonstration item type when the B2B Advanced Shipment Notice event is received from an external source. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.asn
Phase	2
Status	Enabled
Rule Data	Key
Workflow Item Type	WFEVDEME
Workflow Process Name	RCVPOCRT
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

B2B Invoice Event

This event is sent from the Supplier system to the Buyer system as an invoice after the B2B Purchase Order event is received and processed.

Internal Name
`demo.oracle.wf.b2b.po.invoice`

Status
Enabled

Generate Function
None

Owner Name
Oracle Workflow

Owner Tag
FND

Customization Level
Limit

Oracle Workflow provides two default subscriptions to the B2B Invoice event. The first subscription adds a correlation ID to the event message when the B2B Invoice event is received from an external source. The correlation ID consists of the prefix `PO` followed by the event key (the order number). This subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.invoice
Phase	1
Status	Enabled
Rule Data	Key
Rule Function	wf_eventdemo.derivecorrelationid
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

The second subscription sends the event message to the Buyer: Top Level PO process in the Event System Demonstration item type when the B2B Invoice event is received from an external source. The subscription is enabled by default. The following table lists the properties defined for this subscription.

Subscription Properties

Subscription Property	Value
System	*<local system>*
Source Type	External
Event Filter	demo.oracle.wf.b2b.po.invoice
Phase	2
Status	Enabled
Rule Data	Key
Workflow Item Type	WFEVDEME
Workflow Process Name	RCVPOCRT
Owner Name	Oracle Workflow
Owner Tag	FND
Customization Level	Limit

11

Error Handling

This chapter describes how Oracle Workflow handles errors in workflow processes and event subscription processing.

This chapter covers the following topics:

- Error Handling

Error Handling

Oracle Workflow provides default error handling for both workflow processes and event subscription processing. The default handling is defined by error processes provided in a special item type called System: Error, and, for events, by a special subscription to the Unexpected event with a source type of Error. You can also choose to define your own custom error handling by creating custom error processes and subscriptions.

Related Topics

Error Handling for Workflow Processes, page 11-1

Error Handling for Event Subscription Processing, page 11-3

System: Error Item Type and Item Attributes, page 11-5

Default Error Process, page 11-6

Retry-only Process, page 11-9

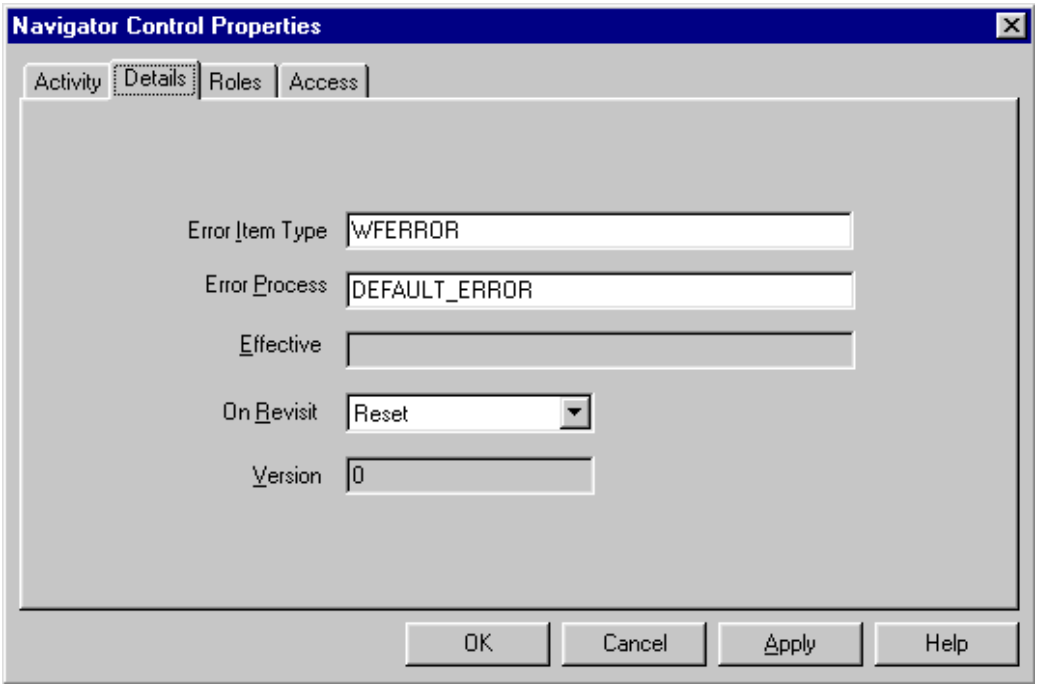
Default Event Error Process, page 11-11

Error Handling for Workflow Processes

Errors that occur during workflow execution cannot be directly returned to the caller, since the caller generally does not know how to respond to the error. In fact, the caller may be a background engine with no human operator. Instead, Oracle Workflow lets you define the processing you want to occur in case of an error by specifying an error handling process when you create your workflow process in Oracle Workflow Builder.

At design time, you can assign an error handling process for a process, function, or event activity in the activity's Details property page. You must specify the internal names of both the item type that owns the error handling process and the error handling process itself. See: To Define Optional Activity Details, page 3-67.

Details Property Page



Oracle Workflow provides a special item type called System: Error, which contains error processes called Default Error process and Retry-only process that you can use for generic error handling in any of your processes. However, you cannot modify the error processes in the System: Error item type. If you want to incorporate functionality that is not available in these error processes, you should create your own custom error handling process in your own item type. An error process can branch based on error codes, send notifications, and attempt to deal with the error using automated rules for resetting, retrying, or skipping the failed activity.

> **Note:** Rather than relying on an error process to handle errors due to specific business rule incompatibilities, you should try to model those situations into your workflow process definition. For example, if a function activity can potentially encounter an error because a business prerequisite is not met, you might model your process to send a notification to an appropriate role to correct that situation if it occurs, so that the workflow process can progress forward. If you do not model this situation into your workflow process, and instead rely on the error to activate an error process, the entire workflow process will have an `ERROR` status and will halt until a workflow administrator handles the error.

At runtime, the Workflow Engine traps errors produced by function activities by setting a savepoint before each function activity. If an activity produces an unhandled exception, the engine performs a rollback to the savepoint, and sets the activity to the `ERROR` status.

> **Note:** For this reason, you should never commit within the PL/SQL procedure of a function activity. The Workflow Engine never issues a commit as it is the responsibility of the calling application to commit.

The Workflow Engine then attempts to locate an error process to run by starting with the activity which caused the error and then checking each parent process activity until an associated error process is located. If the Workflow Engine finds an error process, it launches that process to handle the error. If no error process is assigned for the running process, the Workflow Engine performs a rollback to the savepoint and sets the activity to the `ERROR` status, but no error process can be launched.

> **Note:** Both the error item type and the error process must be specified for an error process to be launched.

Customizing Error Notifications for an Item Type

The Default Error process and the Retry-only process both send notifications to inform an administrator that an error occurred. If you assign one of these processes as the error process for your own process activity, you can control the behavior of the error notifications through item type attributes that you define in your own item type. In this way you can customize your error handling without modifying the predefined error processes.

By default, the Default Error process and the Retry-only process send error notifications to the `SYSADMIN` role. You can optionally specify another role as the recipient of error notifications for your item type by creating an item type attribute named `WF_ADMINISTRATOR` in your item type and setting the value of this attribute to the role that you want to notify. If your item type includes the `WF_ADMINISTRATOR` attribute, the Default Error process and the Retry-only process send any error notifications to the role you specify in that attribute, overriding the default.

For example, suppose you have a requisition approval workflow and you want the purchasing administrator, not the system administrator, to resolve any problems that arise from this workflow. You can define an item attribute called `WF_ADMINISTRATOR` in the item type that owns your requisition approval workflow and set the `WF_ADMINISTRATOR` attribute to the purchasing administrator's role, which may be `PO_ADMIN`.

You can also optionally use an item type attribute in your own item type to control whether the error notification activity in the Default Error process or the Retry-only process times out. Create an item type attribute of type number named `ERROR_TIMEOUT`, and set the value of this attribute to the timeout interval you want, specified in minutes. The Workflow Engine interprets the timeout value of this attribute as a relative offset from the begin date of the activity. If the `ERROR_TIMEOUT` attribute contains a null value, a value of zero, or is not defined at all, the error notification activity does not time out.

Related Topics

System:Error Item Type and Item Attributes, page 11-5

Default Error Process, page 11-6

Retry-only Process, page 11-9

Error Handling for Event Subscription Processing

The Event Manager uses the status codes returned by event subscription rule functions to monitor the status of subscription processing for an event. If a rule function returns a status code of `WARNING` or `ERROR`, indicating that a warning condition or an error occurred during subscription processing, the Event Manager places the event message

on a standard error queue. For subscription processing in the database, the event message is placed on the standard WF_ERROR queue associated with the WF_ERROR agent. In Oracle Applications, for subscription processing in the middle tier, the event message is placed on the standard WF_JAVA_ERROR queue associated with the WF_JAVA_ERROR agent.

For a `WARNING` status, the Event Manager then continues subscription processing for the event. For an `ERROR` status, the Event Manager halts subscription processing for the event and rolls back any subscriptions already executed for the event.

> **Note:** All subscription processing is rolled back in the case of an error in order to allow the system administrator to handle the error by retrying the event, if appropriate. In this way, Oracle Workflow ensures that no subscription is duplicated when subscription processing is restarted.

> **Note:** If a rule function raises an exception, the Event Manager rolls back all subscription processing for the event and raises the error to the calling application. In this case the event message is not placed on the WF_ERROR queue.
>
> If you want the calling application to be aware of the error, you should raise an exception in your rule function rather than returning an `ERROR` status, as the `ERROR` status is handled within Oracle Workflow and is not raised to the calling application.

The WF_ERROR agent is a standard agent for error handling in the database that is automatically defined on the local system when you install Oracle Workflow. In Oracle Applications, Oracle Workflow also provides the standard WF_JAVA_ERROR agent for error handling in the middle tier. Oracle Workflow provides agent listeners to monitor these agents. When a listener dequeues an event message from the WF_ERROR queue or the WF_JAVA_ERROR queue, the message is assigned a source type of Error. The Event Manager then searches for and executes any subscriptions by the local system to that event or to the Any event with the source type Error. If no subscriptions are found, the Event Manager executes any subscriptions by the local system to the Unexpected event with the source type Error.

Oracle Workflow provides one predefined subscription to the Unexpected event with the source type Error. This subscription performs the default error handling for any errored event for which you have not defined a custom Error subscription. The subscription sends the event message to the Default Event Error process in the System: Error item type.

> **Important:** You must not change or disable the definition of the Unexpected event or of the predefined Error subscription to that event. If you do, the Event Manager will not be able to perform default error handling for event and subscription processing.

The Default Event Error process sends a notification to the system administrator. For a warning condition, no response is required. For an error, the process allows the system administrator to abort or retry the event subscription processing.

You can set up custom error handling for a particular event by defining a subscription to that event with a source type of Error and specifying the custom processing you want to execute as the subscription action. In this case, the Event Manager will not perform

the default error handling, since the errored event will no longer be an unexpected event. Instead, your custom error handling will replace the default error handling.

Unexpected Events

Oracle Workflow also uses the Default Event Error process to handle unexpected events. If an event is received from an external source, but the local system does not have any subscriptions to that event, the Event Manager automatically searches for subscriptions to the Unexpected event with the source type External. Oracle Workflow provides a predefined External subscription to the Unexpected event that sends the event message to the Default Event Error process.

The Default Event Error process notifies the system administrator of the unexpected event and allows the system administrator to abort or retry the event subscription processing. For example, the system administrator can create a subscription to handle the event and then re-enqueue the event message to trigger the new subscription.

> **Note:** Oracle Workflow also provides a predefined Local subscription to the Unexpected event that sends the event message to the Default Event Error process when there are no subscriptions to a locally raised event. However, this subscription is disabled by default, because many local events may be raised to which you do not want to subscribe. If you want to enable this subscription, be careful to consider all the events that can be raised on your local system and trigger the subscription.

Related Topics

Unexpected Event, page 9-17

Default Event Error Process, page 11-11

System: Error Item Type and Item Attributes

To view the details of the System: Error item type, choose Open from the File menu in the Oracle Workflow Builder, then connect to the database and select the System: Error item type or connect to a file called `wferror.wft` in the `<ORACLE_HOME>\wf\Data\<Language>` directory.

The System: Error item type contains the following item attributes:

- Error Activity ID
- Error Activity Label
- Error Assigned User
- Error Item Type
- Error Item Key
- Error User Key
- Error Message
- Error Name
- Error Notification ID
- Error Result Code
- Error Stack

- Error Monitor URL
- Timeout Value
- Event Name
- Event Details
- Event Message
- Event Key
- Event Data URL
- Event Subscription
- Error Type

These item attributes are referenced by the function, notification, and event activities that make up the error processes called Default Error Process, Retry-only, and Default Event Error Process.

> **Important:** If you create a custom error handling process in your own item type, Oracle Workflow automatically sets the above item attributes when it calls your error handling process. If these item attributes do not already exist in your process, Oracle Workflow creates them at runtime. However, if you want to reference these item attributes in your error handling process, such as in a message, you must first create them as item attributes in your process's item type at design time using Oracle Workflow Builder.

Related Topics

Default Error Process, page 11-6

Retry-only Process, page 11-9

Default Event Error Process, page 11-11

Default Error Process

`DEFAULT_ERROR` is the internal name of the Default Error Process. The purpose of this error handling process is to:

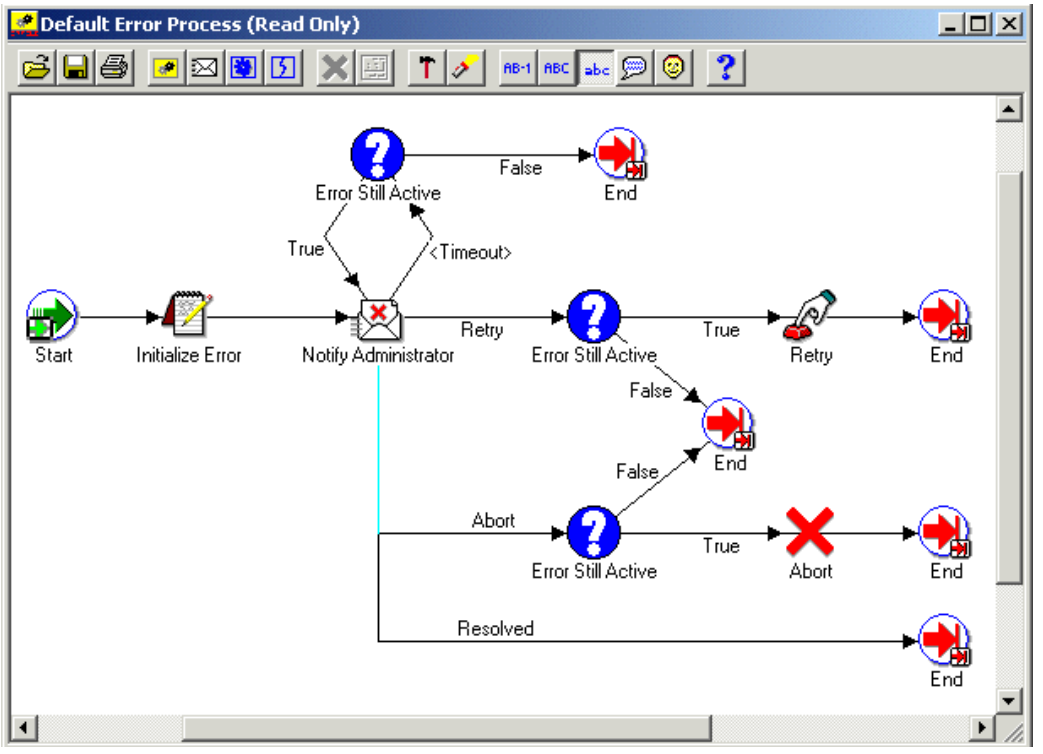
- send an administrator a notification when an error occurs in a process
- provide information to the administrator about the error
- allow the administrator to abort the process, retry the errored activity, or resolve the problem that caused the error to occur

The process automatically terminates when the error is no longer active.

Although you cannot edit the Default Error Process, the process is flexible enough for you to customize its behavior. You can define two item type attributes called `WF_ADMINISTRATOR` and `ERROR_TIMEOUT` in your item type that calls the Default Error Process to control the error processing that is performed.

- `WF_ADMINISTRATOR` - Specify the role to which Oracle Workflow sends the error notification. The default is the `SYSADMIN` role.
- `ERROR_TIMEOUT` - Specify whether the error notification times out.

Default Error Process Diagram



'Initialize Error' Function Activity

The Initialize Error activity calls a PL/SQL procedure named *WF_STANDARD.INITIALIZEERRORS*. This procedure determines if the item type of the errored process has an item type attribute defined with an internal name of `WF_ADMINISTRATOR`. If it does, it sets the performer of the subsequent notification activity, Notify Administrator, to the role stored in `WF_ADMINISTRATOR`. If it does not, the subsequent notification activity remains set to the default performer, `SYSADMIN`.

By checking for an item attribute called `WF_ADMINISTRATOR` in your errored process's item type, the Initialize Error activity lets you specify who you want a notification to be sent to in the case of an error in your specific process without modifying the error process.

'Notify Administrator' Notification Activity

The Notify Administrator activity sends the Default Retry Error message to a performer (either `SYSADMIN` or whatever role is stored in your item type's `WF_ADMINISTRATOR` item attribute). The message indicates that an error has occurred in the specified process and that a response is needed. The response options and their resulting actions are:

- Abort the process - executes the Error Still Active activity to verify if the error is still present and if it is, calls the Abort function activity and ends the default error process.
- Retry the process - executes the Error Still Active activity to verify if the error is still present and if it is, calls the Retry function activity and ends the default error process.
- Resolved the process - ends the default error process because you addressed the errored process directly through some external means or using the embedded URL link to the Workflow Monitor.

> **Note:** The notification message's embedded monitor URL displays the process in error in the Workflow Monitor with full administrator privileges. You can perform actions such as retrying, skipping, or rolling back part of your process to resolve the error.

The subject and body of the Default Retry Error message are as follows:

```
Subject: Error in Workflow &ERROR_ITEM_TYPE/&ERROR_ITEM_KEY

&ERROR_MESSAGE

Body: An Error occurred in the following Workflow.

Item Type = &ERROR_ITEM_TYPE
Item Key = &ERROR_ITEM_KEY
User Key =&ERROR_USER_KEY

Error Name = &ERROR_NAME
Error Message = &ERROR_MESSAGE
Error Stack = &ERROR_STACK

Activity Id = &ERROR_ACTIVITY_ID
Activity Label = &ERROR_ACTIVITY_LABEL
Result Code = &ERROR_RESULT_CODE
Notification Id = &ERROR_NOTIFICATION_ID
Assigned User = &ERROR_ASSIGNED_USER

&MONITOR
```

The Notify Administrator notification activity has a dynamic timeout value assigned to it. It checks the item type of the errored process for an item type attribute whose internal name is `ERROR_TIMEOUT`. `ERROR_TIMEOUT` must be an attribute of type number. The Workflow Engine interprets the value of this attribute as a relative offset from the begin date of the activity, in the unit of minutes, to determine the timeout value of Notify Administrator. If `ERROR_TIMEOUT` contains a null value, a value of zero, or is not defined at all, then Notify Administrator has no timeout.

'Error Still Active' Function Activity

The Workflow Engine initiates the Error Still Active function activity if the Notify Administrator activity times out or returns Abort or Retry as a result.

The Error Still Active activity calls a PL/SQL procedure called *WF_STANDARD. CHECKERRORACTIVE*. The purpose of the Error Still Active activity is to determine whether the errored process is still in error before continuing with the error handling. If it is, Error Still Active returns `TRUE` and the Workflow Engine takes the appropriate transition to either send another notification or abort or retry the errored process. If the errored process is no longer in error, this activity returns `FALSE` and the error handling process ends, as modelled in the process diagram.

'Retry' Function Activity

The Retry function activity executes the PL/SQL procedure *WF_STANDARD. RESETERROR* to clear the activity that was in error and run it again. This procedure calls the *WF_ENGINE.HandleError* API to rerun the activity.

'Abort' Function Activity

The Abort function activity executes the PL/SQL procedure *WF_STANDARD.ABORT PROCESS*, which in turn calls the *WF_ENGINE.AbortProcess* API to abort the process that encountered the error.

Related Topics

Workflow Core APIs, *Oracle Workflow API Reference*

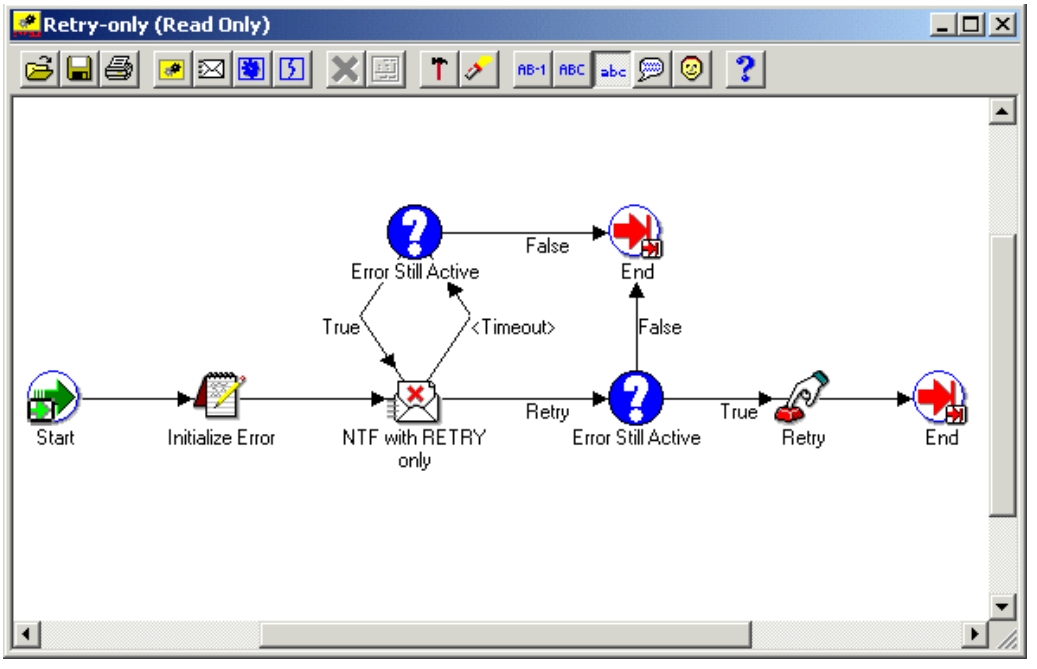
Retry-only Process

`RETRY_ONLY` is the internal name of the Retry-only error process. The purpose of this error handling process is to alert an administrator when an error occurs in a process and prompt the administrator to retry the process in error. The process automatically terminates when the error is no longer active.

Although you cannot edit the Retry-only process, the process is flexible enough for you to customize its behavior. You can define two item type attributes called `WF_ADMINISTRATOR` and `ERROR_TIMEOUT` in your item type that calls the Retry-only process to control the error processing that is performed.

- `WF_ADMINISTRATOR` - Specify the role to which Oracle Workflow sends the error notification. The default is the `SYSADMIN` role.
- `ERROR_TIMEOUT` - Specify whether the error notification times out.

Retry-only Process Diagram



'Initialize Error' Function Activity

The Initialize Error activity calls a PL/SQL procedure named *WF_STANDARD.INITIALIZEERRORS*. This procedure determines if the item type of the errored process has an item type attribute defined with an internal name of `WF_ADMINISTRATOR`. If it does, it sets the performer of the subsequent notification activity, NTF with RETRY Only, to the

role stored in WF_ADMINISTRATOR. If it does not, the subsequent notification activity remains set to the default performer, SYSADMIN.

By checking for an item attribute called WF_ADMINISTRATOR in your errored process' item type, the Initialize Error activity lets you specify who you want a notification to be sent to in the case of an error in your specific process without modifying the error process.

'NTF with RETRY Only' Notification Activity

The NTF with RETRY Only activity sends the Retry As Only Option message to a performer (SYSADMIN or whatever role is stored in your item type's WF_ADMINISTRATOR item attribute). The message indicates that an error has occurred in the specified process and prompts the administrator to retry the activity that errored. The error process then transitions to the Retry function activity and ends the Retry-only error process.

> **Note:** The notification message's embedded URL link displays the process in error in the Workflow Monitor with full administrator privileges. You can perform actions such as retrying, skipping, or rolling back part of your process within the Workflow Monitor to resolve the error.

The subject and body of the Retry As Only Option message are as follows:

```
Subject: Error in Workflow &ERROR_ITEM_TYPE/&ERROR_ITEM_KEY

&ERROR_MESSAGE

Body: An Error occurred in the following Workflow.

Item Type = &ERROR_ITEM_TYPE
Item Key = &ERROR_ITEM_KEY
User Key =&ERROR_USER_KEY

Error Name = &ERROR_NAME
Error Message = &ERROR_MESSAGE
Error Stack = &ERROR_STACK

Activity Id = &ERROR_ACTIVITY_ID
Activity Label = &ERROR_ACTIVITY_LABEL
Result Code = &ERROR_RESULT_CODE
Notification Id = &ERROR_NOTIFICATION_ID
Assigned User = &ERROR_ASSIGNED_USER

&MONITOR
```

The NTF with RETRY Only notification activity has a dynamic timeout value assigned to it. It checks the item type of the process in error for an item attribute that has an internal name called ERROR_TIMEOUT. ERROR_TIMEOUT must be an attribute of type number. The Workflow Engine interprets the timeout value of this attribute as a relative offset from the begin date of the activity, in the unit of minutes. If ERROR_TIMEOUT contains a null value, a value of zero, or is not defined at all, then NTF with RETRY Only has no timeout.

'Error Still Active' Function Activity

The Workflow Engine initiates the Error Still Active function activity if the NTF with RETRY Only activity times out.

The Error Still Active activity calls a PL/SQL procedure called *WF_STANDARD.CHECKERRORACTIVE*. The purpose of the Error Still Active activity is to determine whether the errored process is still in error before continuing with the error handling. If it is, Error Still Active returns `TRUE` and the Workflow Engine transitions back to the NTF with RETRY Only notification activity to send another notification to the administrator. If the errored process is no longer in error, this activity returns `FALSE` and the error handling process ends, as modelled in the process diagram.

'Retry' Function Activity

The Retry function activity executes the PL/SQL procedure *WF_STANDARD.RESETERROR* to clear the activity that was in error and run it again. This procedure calls the *WF_ENGINE.HandleError* API to rerun the activity.

Related Topics

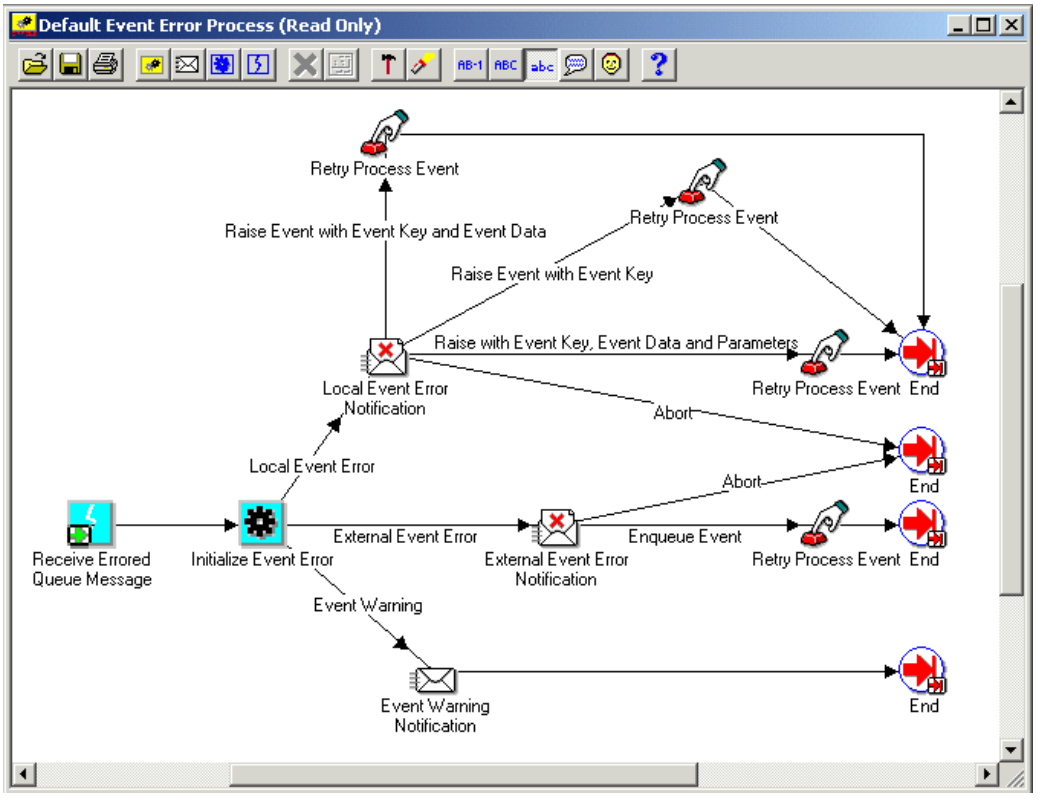
Workflow Core APIs, *Oracle Workflow API Reference*

Default Event Error Process

`DEFAULT_EVENT_ERROR` is the internal name of the Default Event Error Process for the Business Event System. The purpose of this error handling process is to:

- send an administrator a notification when an error or warning condition occurs during event subscription processing
- provide information to the administrator about the error
- allow the administrator to abort or retry the event subscription processing

Default Event Error Process Diagram



'Receive Errored Queue Message' Event Activity

The Receive Errored Queue Message activity receives an event message that has encountered an error or warning condition during subscription processing. For instance, if a subscription rule function returns an `ERROR` or `WARNING` status code, or if an unexpected event is received, the Event Manager executes default subscriptions that send the event message to the Default Event Error process. See: Predefined Workflow Events, page 9-1.

The Receive Errored Queue Message activity stores the event name, event key, and the complete event message in item type attributes.

'Initialize Event Error' Function Activity

The Initialize Error activity calls a PL/SQL procedure named *WF_STANDARD.INITIALIZEEVENTERROR*. This procedure determines the error type.

- Event Warning - A warning condition occurred, but subscription processing continued. For this error type, the Workflow Engine sends the Event Warning Notification.
- External Event Error - An error occurred to halt subscription processing for an event received from an external source. For this error type, the Workflow Engine sends the External Event Error Notification.
- Local Event Error - An error occurred to halt subscription processing for an event raised on the local system. For this error type, the Workflow Engine sends the Local Event Error Notification.

'Event Warning Notification' Activity

The Workflow Engine initiates the Event Warning Notification activity when the errored event has an error type of Event Warning. The activity sends the Default Event Warning message to the system administrator to indicate that a warning condition has occurred during subscription processing. This message is an FYI message and does not require a response.

The subject and body of the Default Event Warning message are as follows:

```
Subject: Event WARNING : &EVENT_NAME / &EVENT_KEY

Body: A Warning occurred in the following Event
Subscription: &EVENT_SUBSCRIPTION

Event Error Name: &ERROR_NAME
Event Error Message: &ERROR_MESSAGE
Event Error Stack: &ERROR_STACK

Event Data: &EVENT_DATA_URL

Other Event Details: &EVENT_DETAILS
```

'External Event Error Notification' Activity

The Workflow Engine initiates the External Event Error Notification activity when the errored event has an error type of External Event Error. The activity sends the Default External Event Error message to the system administrator. This message indicates that an error has occurred during subscription processing for an event received from an external source, and that a response is needed. The response options and their resulting actions are:

- Abort - aborts subscription processing and ends the Default Event Error process. For example, if the event data contained in an event message is corrupted, the system administrator can abort subscription processing on that event message.
- Enqueue Event - executes the Retry Process Event activity to enqueue the event message back onto the queue where it was originally received, and ends the Default Event Error process. The event message is enqueued with a priority of `-1` so that it will be the first message to be dequeued the next time the listener runs.

  The system administrator can attempt to correct the error before re-enqueuing the event. For example, the system administrator can create a subscription to handle an unexpected event and then re-enqueue the event message to trigger the new subscription.

The subject and body of the Default External Event Error message are as follows:

```
Subject: External Event &ERROR_TYPE : &EVENT_NAME / &EVENT_KEY

Body: An Error occurred in the following Event
Subscription: &EVENT_SUBSCRIPTION

Event Error Name: &ERROR_NAME
Event Error Message: &ERROR_MESSAGE
Event Error Stack: &ERROR_STACK

Event Data: &EVENT_DATA_URL

Other Event Details: &EVENT_DETAILS
```

'Local Event Error Notification' Activity

The Workflow Engine initiates the Local Event Error Notification activity when the errored event has an error type of Local Event Error. The activity sends the Default Local Event Error message to the system administrator. This message indicates that an error has occurred during subscription processing for an event raised on the local system, and that a response is needed. The response options and their resulting actions are:

- Abort - aborts subscription processing and ends the Default Event Error process.
- Raise Event with Event Key - executes the Retry Process Event activity to reraise the event with only the event name and event key, and ends the Default Event Error process.
- Raise Event with Event Key and Event Data - executes the Retry Process Event activity to reraise the event with only the event name, event key, and event data, and ends the Default Event Error process.
- Raise Event with Event Key, Event Data and Parameters - executes the Retry Process Event activity to reraise the event with the event name, event key, event data, and parameters, and ends the Default Event Error process.

The system administrator can choose the level of information to provide to the Event Manager when reraising the event. For example, if an error exists in the event data that was originally provided, the event can be reraised with only the event name and the event key, forcing the Event Manager to regenerate the event data using the event's Generate function.

The system administrator can also attempt to correct the error before reraising the event.

The subject and body of the Default Local Event Error message are as follows:

```
Subject: Local Event &ERROR_TYPE : &EVENT_NAME / &EVENT_KEY

Body: An Error occurred in the following Event
Subscription: &EVENT_SUBSCRIPTION

Event Error Name: &ERROR_NAME
Event Error Message: &ERROR_MESSAGE
Event Error Stack: &ERROR_STACK

Event Data: &EVENT_DATA_URL

Other Event Details: &EVENT_DETAILS
```

'Retry Process Event' Function Activity

The Retry Process Event activity executes the PL/SQL procedure *WF_STANDARD.RETRYRAISE*. Depending on the notification response selected by the system administrator, this procedure either re-enqueues or reraises an errored event. The responses that can initiate the Retry Process Event activity and their resulting actions are:

- Enqueue Event - enqueues an errored external event message back onto the queue where it was originally received. The event message is enqueued with a priority of `-1` so that it will be the first message to be dequeued the next time the listener runs.
- Raise Event with Event Key - reraises a local errored event with only the event name and event key.
- Raise Event with Event Key and Event Data - reraises a local errored event with only the event name, event key, and event data.
- Raise Event with Event Key, Event Data and Parameters - reraises a local errored event with the event name, event key, event data, and parameters.

Related Topics

Workflow Core APIs, *Oracle Workflow API Reference*

Managing Business Events, page 8-2

A

Oracle Workflow Developer Navigation Paths

This appendix lists the navigation paths to Oracle Workflow developer Web pages in the seeded Oracle Workflow responsibilities for Oracle Applications.

This appendix covers the following topics:

- Oracle Workflow Developer Navigation Paths

Oracle Workflow Developer Navigation Paths

This table shows the navigation paths to Oracle Workflow developer Web pages in the seeded Oracle Workflow responsibilities for Oracle Applications. Your system administrator may have customized the responsibilities, menus, and navigation paths in your installation.

Oracle Workflow Developer Navigation Paths

Web Page	Standard Navigation Path
Developer Studio, page 7-2	Workflow Administrator Web Applications: Developer Studio *or* Workflow Administrator Web (New): Developer Studio *or* Workflow Administrator Event Manager: Developer Studio
Event Manager, page 8-49	Workflow Administrator Web Applications: Business Events *or* Workflow Administrator Web (New): Business Events *or* Workflow Administrator Event Manager: Business Events

Note: The Workflow Administrator Web (New) responsibility is provided for reference to highlight the Oracle Applications Framework-based functionality that has been added in Oracle Workflow. The same functionality is also available on the previously existing responsibilities, Workflow Administrator Web Applications and Workflow Administrator Event Manager.

B

Oracle Workflow Builder Menus and Toolbars

This appendix provides a description of the menus and toolbars in Oracle Workflow Builder.

This appendix covers the following topics:

- Oracle Workflow Builder Menus
- Oracle Workflow Builder Toolbars

Oracle Workflow Builder Menus

The Oracle Workflow Builder main menu bar includes the following menus:

- File, page B-1
- Edit, page B-2
- View, page B-3
- Window, page B-4
- Help, page B-4

File Menu

The File menu lets you perform several actions.

- New - Creates a new workspace for you to define an item type.
- Quick Start Wizard - Creates a framework from which you can begin designing a workflow process definition. See: Quick Start Wizard Overview, page 2-14.
- Open... - Opens a data store by prompting you to connect to a database or a file. See: Opening and Saving Item Types, page 2-9.
- Close Store - Closes the selected data store. This menu option is available only if the Navigator is the active window.
- Save - Saves changes to the currently connected database or file. See: Opening and Saving Item Types, page 2-9.
- Save As - Save changes to the file or database you specify with an optional effective date.
- Create Shortcut - Creates a shortcut icon on your desktop of the current Oracle Workflow Builder session. Prompts for a shortcut name. The shortcut runs Oracle Workflow Builder and automatically connects to the data store that was selected at

the time you created the shortcut, loading in the item types and opening the process windows that were loaded and open at the time. If the data store is a database, the shortcut prompts for the database password before starting Oracle Workflow Builder. This feature is available only when you run Oracle Workflow Builder in Microsoft Windows 98 or Windows NT 4.0 or higher. Earlier versions of Microsoft Windows NT do not support the concept of shortcuts. See: Creating a Shortcut Icon for a Workflow Process, page 4-18.

- Verify - Validates all process definitions in the current data store. Use Refresh to display the latest verification report of the process. See: To Validate a Process, page 4-17.
- Print Diagram - Prints the process diagram displayed in the active process window. See: To Print a Process, page 4-16.
- Show/Hide Item Types... - Displays the Show Item Types window to determine which item types in the current data store to show or hide in the navigator tree.
- Load Roles from Database - Loads the Oracle Workflow directory service roles from the current database store into Oracle Workflow Builder and makes them viewable from the Directory Service branch in the navigator tree as well as from any property page poplist field that references roles. This menu option is available only if the current data store is a database. See: Roles, page 4-19.
- Exit - Exits Oracle Workflow Builder.

Edit Menu

The Edit menu varies depending on whether you select the Navigator window or a process window. The following menu options appear only when you select the Navigator window and apply only to the Navigator window:

- New - Creates a new item type, function activity, process activity, notification activity, event activity, message, lookup type, lookup code, or attribute by displaying its property page(s).
- Copy - Copies the selected object in the navigator tree.
- Paste - Pastes the object from the clipboard into the selected branch of the navigator tree.
- Delete - Deletes the selected object from the navigator tree.
- Find - Displays the Search window so you can enter search criteria to find an object in the navigator tree. See: To Find an Object in the Navigator Tree, page 2-4.
- Find Again - Finds an object in the navigator tree using the same criteria defined previously in the Search window.
- Properties - Shows the property pages of the selected object.
- Process Details - Opens the process window of the selected process activity.
- Move Attribute - Reorders the attributes listed in the current branch of the navigator tree by moving the selected attribute up or down the list.

The following menu options appear only when you select a process window and apply only to the selected process window:

- Delete Selection - Deletes the selected object(s) from the process window.

- Properties - Shows the property pages of the selected activity node.
- Copy Diagram - Copies the process diagram displayed in the active process window to the clipboard. See: To Copy a Process Diagram to the Clipboard, page 4-17.

View Menu

The View menu lets you alter the display of Oracle Workflow Builder.

- Font - Displays the Fonts property page. Use the property page to change the font settings of the text that appear in the Navigator and process windows. Changes apply to all future sessions of Oracle Workflow Builder. See: Modifying Fonts in Oracle Workflow Builder: , page 4-18.
- Log -> Show - Toggles between displaying and hiding the Log window. The Message Log window displays messages from the Workflow Builder that are not error-related.
- Log -> Detailed - Toggles the debug mode of Oracle Workflow Builder on and off. When you check Detailed, you turn the debug mode on and cause Oracle Workflow Builder to write more extensive messages to the Log window. You should not check Detailed unless instructed to do so by your Oracle customer support representative, as this mode significantly slows down the Oracle Workflow Builder.
- Log -> To File - Writes all future content of the Message Log window to a file. Select Log Show from the View menu to determine the location and name of the log file.
- Log -> Bring to Front - Brings the Message Log window to the front as the active window.
- Grid Snap - Toggles grid snap on or off for all process windows.
- Show Label in Designer submenu - A submenu of options that let you control the information displayed in an activity's label. Choose either Instance Label, Internal Name, Display Name, Performer, or Comment.
- Show Label in Designer -> Instance Label - Uses the node label as the label for each activity node in a process diagram. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.
- Show Label in Designer -> Internal Name - Uses the internal name of an activity as the label for each activity node in a process diagram. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.
- Show Label in Designer -> Display Name - Uses the display name of an activity as the label for each activity node in a process diagram. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.
- Show Label in Designer -> Performer - Uses the activity's performer as the label for each activity node in a process diagram. Function and process activities that do not have performers do not have a label. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.
- Show Label -> Comment - Uses the activity's comment as the label for each activity node in a process diagram. Activities that do not have a comment do not have a label. This setting persists for all process diagrams and for all sessions of Oracle Workflow Builder until you specifically make a change.

- Developer Mode - Toggles the display between standard presentation mode and developer mode. In developer mode, all icons revert to the default icon for the specific object type/subtype, subprocess icons are distinct from top level process icons, and in the navigator tree, objects are shown and sorted by internal name. Note that attributes are shown by internal name but are not sorted.

If the Navigator window is the active window, then the following menu option also appears:

- Split Window - Splits the Navigator window horizontally or vertically.

If a process window is the active window, then the following menu options also appear:

- Overview - Displays the process Overview window. See: To Display a Process Overview, page 4-15.
- Show Process in Navigator - For the current process displayed in the process diagram window, this menu option locates its corresponding process activity in the Navigator window.
- Show Overlay Image - Toggles the display to either show or hide the overlay image for an icon, if it has one. For example, the Start and End activities in a process have a green arrow and red arrow overlay image, respectively.

Window Menu

The Window menu displays the names of all open application windows. Select a window name to make that window active. The following menu choices are also available:

- Cascade - Displays any open windows in a "cascaded" (overlapping) fashion.
- Tile - Displays any open windows in a "tiled" (non-overlapping) fashion.

Help Menu

The Help menu lets you invoke help about using Oracle Workflow.

- Contents - Displays help on how to use Oracle Workflow.
- About Oracle Workflow... - Displays the current version and access level of Oracle Workflow Builder. You can also edit your access level in the Access Level field and apply your change by choosing OK.

Oracle Workflow Builder Toolbars

Oracle Workflow Builder displays a toolbar in both the Navigator window and Process window.

Navigator Toolbar

The Navigator toolbar includes the following buttons which apply only to objects selected from the navigator tree:

- ***New Store***

  Creates a new data store branch in the navigator tree.

- ***Open***

  Displays the Open window to open stored item types from a file or database.

- ***Save***

  Saves any changes in the selected data store to the currently connected database or file. Displays the Open window to let you connect to a database or file if the selected data store is not connected to a database or file.

- ***Delete***

  Deletes the selected object.

- ***Properties***

  Shows the property pages of the selected object.

- ***Copy***

  Copies the selected object.

- ***Paste***

  Pastes the copied object into the current object branch.

- ***Verify***

  Validates the process definition.

- ***Developer Mode***

  Toggles between Developer and Presentation display.

- ***Find***

  Displays the Search window to specify the search criteria to locate an object in the navigator tree.

- ***Quick Start Wizard***

  Runs the Quick Start Wizard to begin creating a workflow process definition.

- ***Help***

  Displays help on how to use Oracle Workflow.

- ***New Object***

  Creates a new object depending on the object branch you select (item type, Processes, Notifications, Functions, Messages, or Lookup Types) by displaying the property page for that object type.

Process Window Toolbar

The process window toolbar includes the following buttons which apply only to objects selected the current process window:

- ***Open***

  Displays the Open window to open stored item types from a file or database.

- ***Save***

  Saves any changes in the selected data store to the currently connected database or file. Displays the Open window to let you connect to a database or file if the selected data store is not connected to a database or file.

- ***Print Diagram***

  Prints the current process diagram.

- ***New Process***

  Displays the process activity node property page for you to create a new process activity.

- ***New Notification***

Displays the notification activity node property page for you to create a new notification activity.

- ***New Function***

  Displays the function activity node property page for you to create a new function activity.

- ***New Event***

  Displays the event activity node property page for you to create a new event activity.

- ***Delete Selection***

  Deletes the selected object.

- ***Properties***

  Shows the property pages of the selected object.

- ***Developer Mode***

  Toggles between Developer and Presentation display.

- ***Find***

  Displays the process Overview window.

- ***Show Instance Labels***

  Displays the instance label of the node as the node activity label in the Process window.

- ***Show Internal Names***

  Displays the internal name of the node as the node activity label in the Process window.

- ***Show Display Names***

Displays the display name of the node as the node activity label in the Process window.

- ***Show Comments***

  Displays the comments of the node as the node activity label in the Process window.

- ***Show Performers***

  Displays the performer of the node as the node activity label in the Process window.

- ***Help***

  Displays help on how to use Oracle Workflow.

C

Oracle Workflow Implementation in Other Oracle Products

This appendix lists embedded workflows and Business Event System implementation in Oracle E-Business Suite and the Oracle technology stack, as well as Oracle's support policy towards the customization of these workflows, events, and subscriptions.

This appendix covers the following topics:

- Predefined Workflows Embedded in Oracle E-Business Suite
- Oracle Workflow Business Event System Implementation in Oracle E-Business Suite
- Oracle Workflow Implementation in the Oracle Technology Stack
- Oracle Workflow Support Policy

Predefined Workflows Embedded in Oracle E-Business Suite

You can use Oracle Workflow to customize the predefined workflow processes listed below. A full description of each workflow is documented in its respective product's User's Guide or Implementation Guide, if one is available.

> **Note:** Some Oracle Applications products use the Account Generator feature to dynamically create accounting flexfield combinations. The Account Generator has generic predefined workflow functions that each Oracle Application product uses in its own predefined Account Generator process. The Account Generator processes for each product are not listed in this section, but are documented in more detail in each respective product's User's Guide. A general discussion of the Account Generator feature is also available in the *Oracle Applications Flexfields Guide*.

Related Topics

Oracle Workflow Support Policy, page C-42

Advanced Planning

Oracle Advanced Supply Chain Planning

Advanced Planning Exception Message Process - Sends notifications to suppliers, customer contacts, or internal personnel that inform them of advanced planning exceptions and lets the recipients initiate appropriate action to correct the planning exception.

Oracle Collaborative Planning

Error Notifications for Excel Import of Forecast/Supply Process - This process sends notifications when you conduct flat file or XML loads to Oracle Collaborative Planning and when loading a supply/demand flat file or loading an XML file adhering to `000_sync_forecast_001.dtd`.

User Defined Exception Workflow Process - In this process, a user-defined custom exception is run and generates a number of results (exceptions). For each generated exception, a notification is sent to the recipient of the exception. The workflow is called for each generated exception. The recipients and subject of the notification can be specified in the Custom Exceptions recipients page.

Supply/Demand Mismatch Process - This process is the Supply Chain Event Manager Exceptions notification workflow.

Oracle Collaborative Planning VMI Replenishment Process - This process is the workflow for the VMI replenishment engine.

Start ASCP Engine Process - This process facilitates the automation of launching an ASCP plan after receiving an order forecast or supply commit from Oracle Collaborative Planning.

DP Receive Forecast Process - This process facilitates receiving forecasts from Oracle Demand Planning into Oracle Collaborative Planning.

Publish Order Forecast Process - This process facilitates the automatic launch of the Publish Order Forecast program from the Planner Workbench.

Publish Supply Commit Process - This process facilitates the automatic launch of the Publish Supply Commit program from the Planner Workbench.

Start Receive Supplier Capacity Process - This process facilitates the automatic start of the Receive Supplier Capacity program when a supply commit is loaded into Oracle Advanced Supply Chain Planning.

Start SCEM Engine Process - This process facilitates the automatic launch of the supply chain event manager when an order forecast or supply commit is loaded into Oracle Collaborative Planning.

Oracle Demand Planning

MSD Demand Planning Cycle Process - The MSD Demand Planning Cycle manages all background processing during a Demand Planning cycle. It is made up of several stages, each of which initiates a specific workflow process to govern a task that is performed during that stage. Each stage is initiated from the Demand Planning Administrator page. Notifying the administrator and user community of relevant processing status as the workflow progresses through its many stages is a central benefit.

The Demand Planning processing cycle is made up of the following five stages, which correspond to processes:

- Downloading data from the Planning Server - Manages the transfer and transformation of data from the Demand Planning Server to the Express Server Demand Planning Engine for analytic processing.
- Populating measures - Runs the statistical forecast and populates each measure that has been defined.
- Distributing to planners - Makes data available to the community of demand planners and notifies each planner that their data is ready.

- Collecting and consolidating data from demand planners - Collects submitted data from demand planners and consolidates data in the Demand Planning Engine. This process can iterate until the date you specify to end the collection period.
- Uploading the consolidated data to the Planning Server - Transfers transformed data back to the Planning Server.

At any time during the Demand Planning processing cycle, the Demand Planning Administrator may choose to run a special process called "Collect Available Submissions Now." This process collects any waiting submissions of data for review prior to the end of the cycle.

The Demand Planning Administrator may choose to run a master workflow process called ODP Master that manages all five stages together by running a concurrent request to launch and manage this master process. This concurrent request is called "Demand Planning Engine Master Workflow Process."

Oracle Global Order Promising

Allocated ATP Process - Sends notification to planners if there was any stealing between different supply sources to satisfy an Order Promising request or if the Order Scheduling process failed.

Applied Technology

Oracle Application Object Library

Oracle Application Object Library provides a set of standard function activities that you can use to incorporate concurrent manager processing into any Oracle Applications workflow process. The standard function activities are associated with the Concurrent Manager Functions item type. See: Concurrent Manager Standard Activities, page 5-17.

Oracle Common Application Components

The following Oracle Common Application Components modules include predefined workflows.

HTML Calendar

JTF Calendar Workflows - Tracks and routes calendar requests to the Calendar Administrator for new group and public calendar approvals and to Group Calendar Administrators for existing group calendar subscription approvals. The process also sends invitations for appointment invitees and attendees.

JTF Task Reminder - Picks up appointment reminders for scheduled appointments. For example, if the user selects "15 minutes Before" from the Remind Me drop-down list while creating an appointment, then that user receives a workflow notification 15 minutes prior to the start of the scheduled meeting.

Escalation Manager

Reactive Escalation Notification Workflow - Sends notifications to inform the escalation owner, the owner's manager, and optionally any customer or employee identified in the escalation document when an escalation document is created, or when certain document attributes such as status, level, owner, and target date are updated.

Business Rule Monitor

Business Rule Monitor Main Process - This process services the Business Rule Monitor looping workflow, which in turn checks all the active business rules and identifies whether any rule has been violated.

Business Rule Monitor Task Process - This process is used only when the Business Rule Monitor Main Process identifies a violated business rule that is related to a Task object. The Business Rule Monitor Task process services the subsequent activities based on the workflow information identified for the rule. For example, if the Escalate a Task (notification only) workflow is selected for a business rule, then a workflow notification will be sent to the person identified in the workflow attributes window.

Business Rule Monitor Service Request Process - This process is similar to the Business Rule Monitor Task process, except that it is used only when the Business Rule Monitor Main Process identifies a violated business rule that is related to a Service Request object.

Business Rule Monitor Defect Process - This process is similar to the Business Rule Monitor Task process, except that it is used only when the Business Rule Monitor Main Process identifies a violated business rule that is related to a Defect object.

Task Manager

Workflow - Task Manager - Task Manager uses the Workflow - Task Manager process to send notifications to inform personnel when tasks are created or changed.

Task Manager supports sending notifications to groups or teams. If a task is assigned to a group or team, then the workflow process will send a separate workflow notification to each member of the group, rather than sending only one notification to the e-mail address listed for the group.

Notifications can be sent either automatically for both the HTML and Forms versions of Task Manager, or manually for the Forms version only.

- Manual Workflow Notifications (Forms only) - After a task is created with owner and assignee information, click the Launch Workflow button in the Tasks window to manually send notifications to the owner and assignees about task creation or updates.
- Automatic Workflow Notifications - Workflow notifications can be automatically sent if one of the following conditions is met:
  - The Auto Notification check box for a task is selected in the Forms-based Tasks window before saving the task.
  - The Notify check box is selected in the HTML-based Create Task window.
  - The Notification check box is selected in the Task Types setup window.

  If automatic workflow notifications are enabled, task workflow notifications are automatically sent in the following cases.

  - When a task is created or deleted, the owner and all assignees will receive notifications.
  - When a task is reassigned to a new owner, the old and new owners will receive notifications.
  - When the assignees for a task are changed, whether by adding, reassigning, or deleting, the owner and the old and new assignees will receive notifications.

- When updates are made to the task status, priority, type, and planned, scheduled, or actual start and end dates, the owner and all assignees will receive notifications.
- When the task owner updates his or her status, the owner and all assignees will receive notifications.

If you do not want to use the default workflow process for Task Manager, you can define a new workflow process using Oracle Workflow and assign your custom workflow to a task type.

Oracle Common Application Calendar

JTF Calendar Workflows - Tracks and routes relevant workflow notifications to an appointment owner when an invitee accepts or rejects an invitation.

JTF Task Reminder - Picks up appointment reminders for scheduled appointments. For example, if the user selects "15 minutes Before" from the Remind Me drop-down list while creating an appointment, then that user receives a workflow notification 15 minutes prior to the start of the scheduled meeting.

Oracle XML Gateway

Oracle XML Gateway leverages Oracle Workflow to publish and subscribe to application business events of interest in order to automatically trigger message creation or consumption. Seeded Workflow functions are provided for use in Workflow processes to interact directly with the XML Gateway Execution Engine to generate outbound or to consume inbound messages. The outbound messages generated by the Execution Engine are made available to the downstream Workflow activity for processing. The Execution Engine consumes the inbound messages passed to it by a Workflow process.

Two item types are delivered with the XML Gateway: the XML Gateway Standard Item Type and the XML Gateway Error Processing Item Type.

XML Gateway Standard Item Type - The XML Gateway Standard Item Type includes the Raise Document Delivery Event, which 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. The functions included with the XML Gateway Standard Item Type are Consume XML Document, Generate XML Document, Generate Trading Partner XML Document, Send Document, Transform XML, and Transaction Delivery Required.

Configure the seeded events and event subscriptions delivered by the Oracle E-Business Suite for pre-built XML messages in support of Business-to-Business or Application-to-Application integration.

XML Gateway Error Processing Item Type - The XML Gateway Error Processing Item Type contains error handling processes to manage errors detected by the Oracle Workflow Business Event System or Oracle XML Gateway. The error processes are: 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, and Error Handling for Outbound Messages.

The XML Gateway Error Processing Item Type supports two event activities: Receive Error and Receive Send Notification Event. The Receive Error event is used by the XML Gateway to indicate that the XML Gateway execution engine has detected an error. The Receive Send Notification Event is used to indicate that the execution engine has identified a need to send a notification for errors related to an inbound process.

Oracle Workflow error handling provides active error notification to the XML Gateway System Administrator or Trading Partner with support for the Workflow retry and reprocess features. The functions provided by the XML Gateway Error Processing Item Type are: ECX Reprocess Inbound, ECX Resend Outbound Message, Get ECX In Error Details, Get ECX Out Error Details, Get System Administrator Role, and Get Trading Partner Role.

For more information, see: *Oracle XML Gateway User's Guide*.

Business Intelligence

Oracle Business Intelligence System

BIS Management by Exceptions Process - This generic workflow process is a template for BIS customers to use as part of their Performance Management Framework. When actual performance does not meet expected performance, this process sends a basic corrective action notification with an embedded report URL. All other processes under the OBIS Corrective Action item type are similar to this generic process.

Process Manufacturing Intelligence

Process Manufacturing Inventory Turns Process - Sends notifications to the designated responsibilities whenever the actual values of the inventory turn do not fall within the targeted values defined in the Inventory Turn Report. The Inventory Turn Report is part of Process Manufacturing BIS.

Communications

Oracle Service Fulfillment Manager

Fulfillment Header, Fulfillment Line, Work Item, Fulfillment Action, and Message Error Workflows - These workflow processes are initiated whenever Oracle Service Fulfillment Manager is invoked to fulfill a network or provisioning request. The header workflow identifies the fulfillment line items and invokes child workflows for each one. Similarly, the line workflows spawn work item workflows, which in turn spawn fulfillment action workflows that actually perform the provisioning tasks. In case of any errors in processing fulfillment messages, the Message error workflows are spawned. Certain types of interaction between Order Management and Installed Base modules use Oracle Service Fulfillment Manager.

Contracts and Service

Oracle Service

Service Request Process - Routes a service request to individuals in the organization for resolution. Customize the process to select and notify service personnel, as well as to transfer and escalate service requests automatically based on your organization's service rules and guidelines.

Service Request Actions and Dispatch Process - Routes a service request action to individuals in the organization for resolution and in addition, notify with instructions, appropriate service personnel who need to be dispatched to a field site. Customize the process to manage, transfer or escalate dispatch requests.

Field Service Dispatch Process - Inserts or updates service request data into the interface table and sends a notification to the field service engineer with dispatch information. This process is used by Oracle Mobile Field Service.

Oracle Service Contracts

Service Request Creation Workflow - An event can be set up against a service contract line so that when the associated condition has been met, this workflow can be used to automatically create a service request. A notification containing details of the service request will also be created, which can be viewed from the inbox of the contracts launchpad.

Electronics Renewals Workflow - This workflow is launched when an extended warranty contract qualifies for the electronic renewals process. It first derives the template set, QA checks the contract, generates a quote (draft contract) letter with associated cover, emails the letter, and updates the renewal status. When the user logs in through self service renewals, if they accept the contract quotation and decide to pay the contract amount using a PO, the workflow sends a notification to the contract administrator. Depending on the administrator's response, it sets the contract to the appropriate status.

Installed Base Transactions Notification Workflow - If the system profile 'OKS: Enable Install Base Integration Messages' is set to '`Yes`', this workflow will send notifications to the user identified in the system profile 'OKS: User name to Send Install Base Messages' during certain transactions. These include when a new item instance is created, when it is terminated, when it is transferred to a new owner, when it is replaced or returned, or when the quantity of items instances are split.

Financial Applications

Oracle Cash Management

CE Statement Transmission Notifications Process - This workflow sends a transmission status e-mail notification to the designated user defined in the Bank Transmission Details window. This workflow process is initiated when you submit the Retrieve Bank Statement program to transmit bank statement files from your bank to your local directory.

CE Exception Report Transmission Notification Process - This workflow sends a transmission status e-mail notification to the designated user defined in the Bank Transmission Details window. This workflow process is initiated when you submit the Retrieve Payment Exceptions program to transmit payment exceptions reports from your bank to your local directory.

Process XML Bank Statement Workflow - This workflow begins once the import process is launched. Notifications are sent to the designated Cash Manager to say whether or not the import process was launched successfully and whether or not the import process was a success.

Oracle Collections

Bankruptcy Status - When an account is flagged as bankrupt, several processes begin:

- Approval is sent to the specialist or manager (HTML user interface) to work the bankruptcy, gather details, and determine if the bankruptcy will be pursued. A bankruptcy strategy is ultimately assigned.
- The delinquency status is set to Bankruptcy for all delinquencies.

- All other items in the case or for that customer (that is, delinquencies) are set to Bankruptcy.
- A No Contact flag is set in TCA for all contacts of the organization, or, in the case of consumer collections, the No Contact flag is set for the guarantor or co-signer.
- For Lease Management Collections only, a notification is sent to the appropriate party who will review and make the necessary changes to stop invoicing.
- For Lease Management Collections only, a Default Bankruptcy Notice of Assignment is sent to the appropriate party.

The Bankruptcy Status workflow is initiated when a collector (Forms user interface) creates a new status of Bankrupt for a single delinquency from the Delinquency tab in Oracle Collections.

Delinquency Current Status Notice - This workflow sends notifications about the status of a delinquency to the agent and the agent's manager when the status changes. For example, when the delinquency is closed (that is, when it is considered "current") or when a new status such as Bankruptcy or Write-Off is created for a delinquency, a notification is sent to describe the change in status.

Collection Delinquent Credit Hold - A collector can request that a credit hold be placed on a customer by checking the corresponding check box on the Delinquency tab. A notification is then sent to the appropriate manager to review and approve or deny the request. Routing to the appropriate manager for approval is done through the reporting hierarchies as defined by Oracle Resource Manager. After the request is approved, the Approval check box on the Delinquency tab is checked so collectors can confirm that their requests have been approved. The actual setting of a credit hold is a manual process, however.

Collection Delinquent Service Hold - A collector can request that a service hold be placed on a customer by checking the corresponding check box on the Delinquency tab. A notification is then sent to the appropriate manager to review and approve or deny the request. Routing to the appropriate manager for approval is done through the reporting hierarchies as defined by Oracle Resource Manager. After the request is approved, the Approval check box on the Delinquency tab is checked so collectors can confirm that their requests have been approved.

Notify a Third Party for Repossession - When a specialist assigns a third party organization to repossess an asset, an e-mail notification is sent to that organization with the details about the asset or assets, repossession, and other information. The specialist assigns the third party through the HTML interface.

Delinquency Status Approval - A collector can recommend that various strategies or statuses should be initiated for a delinquency: repossession, write off or litigation. (Bankruptcy is handled separately.) However, these statuses require review and approval by a manager before the corresponding strategy is initiated. The Delinquency Status Approval workflow is initiated when a collector clicks the New button on the Delinquency tab, selects a new status, enters the details to start the process, and saves the new status. The workflow routes the request according to the resource's hierarchy as defined by the system.

Delinquency Asset Workflow - A specialist can request that an asset valuation be provided for the assets related to a delinquent case. This workflow sends a notification to the appropriate manager to review the request, obtain the information and forward it back to the specialist. The request is initiated on the asset selection window from the Delinquency tab and write off window.

Strategy Fulfillment Mailer - As part of a strategy, a request may be made to the 1-to-1 Fulfillment Server to send e-mail documentation such as a dunning notice, a dunning reminder, copies of delinquent invoices, and so on, to the delinquent party. This workflow manages this request.

Strategy Custom Workflow - If customers want to implement a custom work item as part of strategy in order to use a different implementation from the seeded workflows, they can customize the custom strategy workflow. The custom strategy work item workflow is an example of what parameters are expected to be identified in a custom workflow and what must be identified after the work item is completed.

Collection Strategy Workflow - A strategy is a series of collections work items grouped together to create a collections plan. Each work item may have a workflow as part of the completion process. The Collection Strategy workflow is the main workflow process that drives the completion of all the workflows that are tied to the collections work items for each strategy.

Oracle General Ledger

Journal Approval Process - You can require journal batches to be approved before posting. Create an approval hierarchy and define authorization limits for each user. The Journal Approval process is initiated when you try to post a journal batch. The process automatically routes journals to the appropriate user for approval, based on the approval hierarchy.

AutoAllocations Process - When you generate step-down AutoAllocations, the workflow process initiates the AutoAllocation process and validates and generates the Mass Allocation and Recurring Journal batches that are defined in the AutoAllocation. The workflow process also determines whether journal approval is required for each generated journal batch, submits the batches to the appropriate users for approval if required, and notifies the appropriate users of the approval results. If an error occurs during the AutoAllocation process, the designated user or users can choose to roll back the AutoAllocation process, which reverses any posted journals.

Global Intercompany System - The Global Intercompany System (formerly CENTRA) is an enhanced feature for Release 11*i*, and has been backported to Release 11. It provides an environment for multiple companies to exchange intercompany transactions. The workflow process notifies the receiver company when a sender company initiates an intercompany transaction and requires approval from the receiver, or when the sender company recalls or reverses an intercompany transaction. The workflow process notifies the sender company when a receiver company approves or rejects an intercompany transaction that the sender had initiated. In addition, a threshold amount can be set to limit the volume of notifications. The workflow process is initiated when the sender submits, recalls, or reverses an intercompany transaction, or when the receiver rejects or accepts an intercompany transaction.

Global Consolidation System Cross Instance Data Transfer - You can automate the Global Consolidation System to consolidate data from remote subsidiary ledger database instances to a central consolidation database instance and optionally use workflow notifications to notify users of the cross instance consolidation status. Each notification provides the user with consolidation transfer details including source database name, mapping rule, set of books, group IDs, and concurrent request ID. When a user chooses to automatically run Journal Import or AutoPost on the central consolidation database, the workflow notification cites the status of success or failure of the concurrent request. If the Journal Import or AutoPost process fails, the workflow notification

recommends reviewing the request log on the central consolidation database for further details.

Oracle Grants Accounting

Grants Accounting Workflow Process - The Grants Accounting Workflow process notifies key members that an installment has been activated or that a report is due. The Budget Subprocess notifies the budget approver or award manager that a budget has been submitted for approval.

The workflow process is initiated at the following points:

- installment is activated
- report is due
- budget is submitted
- budget is approved/baselined

Oracle Grants Proposal

Proposal Approval Process - The Proposal Approval Process is initiated when a proposal is submitted for approval.

Notifications are sent to approvers and the workflow process waits for the response from each approver before proceeding to the next approver in the hierarchical proposal approval map.

The proposal is approved if all approvers approve the proposal. The proposal is rejected if any approvers reject it. The person submitting the proposal for approval is notified of the approval status at every stage during the approval process.

Notify Approval Subprocess - The Notify Approval Subprocess is initiated during the Proposal Approval Process when the next approver in the hierarchical approval map is selected.

The Notify Approval Subprocess notifies the approver that a proposal is pending for approval. The approver can approve or reject the proposal.

If the approver fails to approve or reject the proposal within a given time frame, the approver receives periodic reminders. Organizations can set the timeout, which defines the time frame in which the reminders are sent. By default, the timeout is not set.

Notify Proposal Members Process - The Notify Proposal Members Process sends notifications to personnel on the proposal.

Oracle Internet Expenses

Expenses - Oracle Internet Expenses uses the Expenses workflow process to process the manager approval and accounting review of expense reports entered in Internet Expenses. The Expenses process begins when a user submits an expense report, and finishes when an expense report is rejected, or when a manager has approved and accounting has reviewed an expense report. If approved and reviewed, the workflow process makes the expense report available for the Payables Invoice Import program. The Expenses process notifies employees at key event points during the manager approval and accounting review processes.

Credit Cards Workflow - The Credit Cards Workflow process consists of independent workflow processes and notifications that perform various activities. The Credit Cards Workflow process contains the following processes:

- *Aging Credit Card Transactions* - This process notifies employees and managers of outstanding transactions by aging bucket. The process also escalates manager notification.
- *Inform Manager of Inactive Employee Transactions* - This process notifies managers of unsubmitted transactions for employees that are terminated or on temporary leave. It also automatically assigns and unassigns the securing attribute to facilitate transaction submission.
- *Payment to Card Issuer* - This process notifies employees when payments are made to the credit card issuer.
- *Payment to Employee* - This process notifies employees when payments are made to them.
- *Payment to Employee by Check* - This process notifies employees when payments are made to them.
- *Process Invalid Credit Card Transactions* - This process notifies the system administrator when invalid transactions are detected during the import and validation process.
- *Process Unassigned Credit Cards* - This process notifies the system administrator when new credit cards are created. The process also attempts to match new accounts to employees, and can be defined to automatically activate new accounts if a unique match is found.
- *Unapproved Expense Report* - This process notifies managers of unapproved expense reports that contain credit card transactions.
- *Unused Credit Card Transactions* - This process notifies managers and employees of unsubmitted credit card transactions.

Procurement Cards

The following procurement card workflow processes enable your self-service employees to verify and approve procurement card transactions.

AP Procurement Card Employee Verification Workflow Process - The AP Procurement Card Employee Verification Workflow process notifies and confirms procurement card transactions with card holders. This workflow process is initiated when you submit the Distribute Employee Card Transaction Verifications program from Oracle Payables. The process notifies an employee of transactions charged to the employee's procurement card, and optionally requires the employee's manual verification.

AP Procurement Card Manager Approval Transaction Process - The AP Procurement Card Manager Approval Transaction workflow process notifies and confirms verified procurement card transactions with a card holder's manager. This workflow process is initiated when you submit the Distribute Manager Card Transactions Approvals program from Oracle Payables. The AP Procurement Card Employee Verification Workflow process must first complete for transactions before the manager workflow process is used. The process notifies managers, and optionally requires their manual approval, of procurement card transactions incurred by employees.

Oracle Labor Distribution

Effort Report Notification Process - Workflow functionality in Labor Distribution automatically routes effort reports throughout the organization and delivers electronic notifications to users regarding effort reports that require their attention or processes that are completed.

The Effort Report Notification workflow process includes the following subprocesses:

- approval
- notification

The Effort Report Notification workflow process is initiated in Labor Distribution when an effort report is created.

Notification is sent to approvers of the effort report. When the effort report is approved, the effort report is sent to a supervisor for certification. The creator of the effort report can monitor the status of the effort report.

Distribution Adjustment Approval Notification Process - Workflow functionality in Labor Distribution automatically routes distribution adjustments approval notifications throughout the organization and delivers electronic notifications to users regarding distribution adjustments that require their attention or processes that are completed. The process is initiated when a distribution batch is submitted.

Oracle Payables

AP Remittance Advice - When you confirm a payment batch or create a Quick payment, this workflow automatically sends an e-mail to each supplier that has a remittance advice e-mail address defined in the Supplier Sites window.

AP Open Interface Import Process - This workflow automates verification and validation of data in the Payables Open Interface invoice tables. For example, this process can be modified to validate all accounting code combinations in the Payables Open Interface invoice tables. Notification of any invalid code combinations can be sent to a specified user for correction. Optionally the process can be set up to override any invalid code combinations with a designated default value. You can use Oracle Workflow to include additional workflow rules that meet the specific requirements of a business. Once an invoice has passed this process it is ready to be imported into the Oracle Payables application tables. To initiate the Open Interface Import process, submit Payables Open Interface Workflow from the Submit Requests window.

Invoice Approval Workflow - This workflow routes invoices to designated individuals for approval. This workflow uses the approval rules that you define in Oracle Approvals Management (OAM) to determine if an invoice requires approval. If the invoice requires approval, the workflow sequentially asks each approver on the approval list to approve the invoice online. For example, you can define a rule so invoices over $100,000 require CFO approval and then CEO approval.

Process Payment Message Workflow - This workflow creates the XML payment message and sends it to your bank. It then sends notifications to the user who formatted the payment. This workflow also manages the confirmation of payment batches that generate XML payments.

Receive Payment Instruction Error Workflow - If you use the XML payment feature and if you set up Oracle XML Gateway to receive the Show Payment Instruction Error message, then when your bank sends you this message, the Receive Payment Instruction

Error workflow sends the appropriate user a notification that includes the payment instruction errors.

Receive Payment Advice Message Workflow - If you use the XML payment feature and if you set up Oracle XML Gateway to receive the Show Payment Advice XML message, then when your bank sends you this message, the Receive Payment Advice workflow sends a notification that includes the payment advice to the appropriate user.

For more information about related workflows in Oracle Internet Expenses, including expense reports and procurement cards, see Oracle Internet Expenses, page C-10.

Oracle Public Sector Budgeting

Distribute Worksheet Workflow Process - The Distribute Worksheet Workflow Process distributes worksheets and notifies users that a worksheet has been distributed. The process is initiated when distributing a worksheet.

Submit Worksheet Workflow Process - The Submit Worksheet Workflow Process submits worksheets. Based on user-defined parameters, the process performs constraint validations, worksheet operations, copying, and merging. The process moves worksheets from one budget stage to the next and routes the worksheets through an approval process for required approvals. The process also freezes and unfreezes worksheets. Notifications are sent to users who initiate a process and to approvers.

The process is initiated at the following points:

- validating a worksheet constraint
- freezing a worksheet
- unfreezing a worksheet
- moving a worksheet to the next stage
- copying a worksheet
- merging a worksheet
- submitting a worksheet

Distribute Budget Revision Workflow Process - The Distribute Budget Revision Workflow Process distributes budget revisions and notifies users that budget revisions have been distributed. The process is initiated when distributing a budget revision.

Submit Budget Revisions Workflow Process - The Submit Budget Revisions Workflow Process submits budget revisions. Based on user-defined parameters, the process performs constraint validations and other budget revision operations. The Submit Budget Revisions process routes budget revisions through an approval process and updates the status and baseline values for budget revisions. The process also performs funds reservation and posts revisions to General Ledger. The process freezes and unfreezes budget revisions. Notifications are sent to users who initiate a process and to approvers.

The process is initiated at the following points:

- validating a budget revision constraint
- freezing a budget revision
- unfreezing a budget revision
- submitting a budget revision

Oracle Public Sector Financials (International)

ITR Approval - The internal trading approval process is initiated when an internal charge is submitted for approval. The process automatically routes service lines to the appropriate approvers in the creation charge center, based on the approval hierarchy. If successfully approved by the creation charge center, the service lines are routed to the appropriate approvers in the receiving charge center.

EXP Approval - The Exchange Protocol Approval process is initiated when a transmission unit is transmitted. The process sends requests for transmission unit authorization to the appropriate approvers, based on the approval profile. The dialog units that are contained within the transmission unit will be processed based on the approvers' responses.

Contract Commitment Approval Workflow Process - The Contract Commitment Approval Workflow is initiated when a contract commitment is submitted for approval. The process automatically routes the document to the appropriate approver, based on the approval hierarchy setup. The approver can approve or reject the document. If budgetary control is enabled, funds are checked and reserved when the approver approves the document. The person who submitted the document for approval is notified of the approval status at every stage during the approval process.

Dossier Approval Process - This process is initiated when a dossier maintenance transaction is sent into approval. The process sends notifications requesting approval of the dossier maintenance funds transfers to the appropriate approvers, based on the approval hierarchy attached to the dossier type. Based on the approvers' responses, encumbrance journals will be reversed and actual journal entries created to complete the funds transfer.

Oracle Receivables

Credit Memo Request Approval Process - This workflow routes a credit memo request for approval using an organization's internal management hierarchy or approval limits defined in Oracle Receivables. If the request is approved, a credit memo is automatically created in Oracle Receivables. Otherwise, the process notifies the requestor with an explanation of why it was not approved.

You initiate the Credit Memo Request workflow from iReceivables. iReceivables is a Web-based, self-service application that enables registered users to access their Receivables account information using a standard Web browser. When an iReceivables user chooses the Dispute a Bill function, Receivables places the specified amount in dispute and initiates the Credit Memo Request process to route the request for approval.

Document Transfer Message Workflow - This workflow creates an XML invoice document and sends it to your customer. This workflow consists of two item types.

- *AR Transfer Document item type* - In Oracle Receivables, users run the Document Transfer Scheduling and Document Transfer concurrent programs to send XML documents. The Document Transfer Scheduling Program schedules transactions for transmission. The Document Transfer program raises a business event that is subscribed to by a workflow process, which calls Oracle XML Gateway to create and transmit the documents.
- *AR Notification item type* - If the Document Transfer concurrent program encounters technical or transmission errors, then the transmission status changes to Failed and the program sends a workflow notification to the system administrator or Receivables user for exception handling.

In addition, Receivables users can receive via Oracle XML Gateway confirmation messages sent from their customers' payables departments. These messages confirm the

document import statuses. The Receivables program receives the messages and, where necessary, automatically sends workflow notifications to Receivables users for exception handling.

AR Credit Management Application Process Workflow - This workflow manages the collection and analysis of account or prospect credit data, as well as the making and implementation of credit decisions.

The workflow is started when a credit request is generated, either by a credit event, such as an order hold, or by the submission of a credit application. The workflow first tries to automatically complete the credit review process. If, for any reason, a failure occurs in the workflow functions, then the workflow routes the credit analysis to the appropriate credit analyst for action.

If an organization requires that credit recommendations be reviewed and approved by other personnel, then the workflow routes the recommendations through an approval hierarchy.

- If the recommendation is approved by the appropriate personnel, then it is automatically implemented.
- If the recommendation is rejected as it is routed through the approval hierarchy, then notifications are sent to the appropriate personnel and the case folder is updated with the credit decision.

Oracle Treasury

Limits Workflow - Oracle Treasury generates an event limit notification if a deal is recorded that violates one of the rules that you have defined. You can design event assignments to generate notifications for specific limits violations, such as combinations of specific deal types, companies, counterparties, and limit amounts.

Oracle US Federal Financials

Budget Execution Transaction Approval Process - Routes budget transactions through the approval process. Oracle Workflow uses the approval controls and hierarchies defined in the Define Budget Users window within Budget Execution to route documents for approval. The Budget Execution Transaction Approval process is initiated in the following windows by clicking the `Approve...` button:

- Enter Appropriation
- Enter Funds Distribution
- Re-programming Transactions
- Budget Transactions Summary

When a transaction is submitted for approval, the funds checking process is initiated to validate that sufficient funding is available. The transaction cannot be approved if it fails funds checking. A transaction must be approved before it can be transferred to General Ledger.

Higher Education

Oracle Student System

Academic Index - A notification is sent to the subscriber alerting them to a change in the Academic Index. Users can then use this information to make decisions about admission

applications. This workflow is triggered when the Academic Index business event is raised.

Address Change - This workflow is triggered when subscribed from an Oracle Trading Community Architecture business event. A notification provides old and new values for an updated address record. Use an Oracle Trading Community Architecture business event for creation or update of a Party Site and Location to route the Address Change workflow notification to a designated person. A notification is sent to the system administrator by default.

Admission Enforce Single Response Notification - When an offer response is recorded, the Offer Response Status business event is raised. The Admission Enforce Single Response Notification is sent when the offer response status is Accepted and the admission period category is set for single response. The workflow notification is sent to all applications instances for the person that is using an admission period category that is set for single response. This workflow is triggered when the Offer Response Status Change business event is raised and you are enforcing single response.

Admission Offer Response Status Change Notification - When an offer response is recorded, the Offer Response Status business event is raised. The Admission Offer Response Status Change workflow notification is sent every time the offer response status changes for an application instance. This workflow is triggered when the Offer Response Status Change business event is raised.

Admissions Requirements - This workflow notifies applicants that there are additional documents or missing items that are needed to complete the application. This workflow is triggered by running the Admission Requirements concurrent process.

Attendance Submission Notification - This workflow notifies the lead instructor of a unit section that attendance for the unit section is submitted. This workflow is triggered from the Review Attendance self service screen.

Change Grade Request - This workflow notifies the user who initiated a change of grade request and seeks resolution of the change of grade request for final and early final grading periods from the lead instructor of the unit section. It is triggered from the Review Change of Grade self service screen for final and early final periods.

Confirmation of Registration - This workflow is triggered when the Confirmation of Registration business event is raised. This workflow notifies the student, the student's supervisors, and an administrator of the registration of the student for the research candidacy.

Corrections Not Initiated by School - This workflow is triggered by the ISIR Import Process concurrent process. It notifies users when an ISIR correction record is received which the school did not initiate.

DLPNA - This workflow is triggered by the Promissory Note Acknowledgement Process concurrent process, which is part of the request set Direct Loan > Upload Promissory Note Acknowledgement. It is triggered if a Loan's promissory note is rejected by the external processing agency.

Exceed Workload - This workflow allows you to notify staff members when they exceed the expected workload. This workflow is triggered when staff members exceed the expected workload.

Generate New User - This workflow is triggered when a user attempts the new user registration in self service. Self service administrative users can approve or reject the creation of a person.

Grade Submission Notification - This workflow notifies the lead instructor of a unit section that final grades for the unit are submitted. Institutions are allowed to only customize the SELECT_APPROVER process of the Grade Submission and Change of Grade workflows to satisfy their specific needs. The remaining steps of the workflow must remain unchanged. This workflow is triggered from the Review Enter Grade self service screen.

IGS Degree Audit - This workflow delivers the XML document to the trading partner taking advantage of Oracle XML Gateway Callback functionality. The Callback feature allows the messaging system (Oracle Transport Agent) to report the message delivery status back to the workflow process that initiated the message creation. If the delivery fails, a notification is sent to the System Administrator defined in Oracle XML Gateway.

Incomplete Applications - This workflow sends both an e-mail message to the applicant and a notification accessible in the Applicant Home page. An applicant may have multiple incomplete applications. One e-mail message is sent for every time the workflow is initiated by the concurrent manager request, regardless of the number of incomplete applications the applicant may have. The workflow process may be initiated multiple times within a single period. This workflow is triggered by running the Incomplete Applications concurrent process for applicants with incomplete self-service applications.

Inform Instructor about Assessment Item Grades Release to Student - This workflow is triggered by a business event.

Inform Instructor That Unit Section Grades Have Been Submitted - This workflow is triggered by a business event.

Key Program Change - This workflow is triggered when the `oracle.apps.igs.en.prog.keyprim` event is raised. This workflow notifies an administrator when a student's key program changes.

Milestone - This workflow is triggered when the Milestone business event is raised. This workflow notifies the student, the active student's supervisors, and an administrator of additions and changes to milestone information for the research candidacy.

Milestone Notification - This workflow is triggered when the Milestone Notification business event is raised. This workflow notifies the student, the active student's supervisors, and an administrator of additions and changes to milestone notification information for the research candidacy.

Notify Advisor About Advising Group - This workflow notifies each advisor when he or she gets assigned to or removed from an advising group. It is triggered when an advisor gets assigned to or removed from an advising group.

Notify Evaluators About Unevaluated Applications - This workflow allows administrative users to notify application reviewers about applications that are complete and are ready for review.

Notify Student About Advising Group - This workflow notifies each student when he or she gets assigned to or removed from an advising group. It is triggered when a student gets assigned to or removed from an advising group.

Notify Student About Being Placed on an Advising Hold - This workflow notifies each student when he or she has been put on an advising hold for an advising group. It is triggered when a student has been put on an advising hold for a particular advising group.

OSS: Change in Supervisor Attribute - This workflow is triggered when a supervisor is assigned to a research candidacy or when certain attributes of the candidacy's supervision change. These attributes are the Person Number, Start Date, End

Date, Supervised%, Supervisor Type, Organizational Unit, or Replaced Person Number. This workflow notifies students and administrators of supervisors.

OSS: Check Overdue Submission - This workflow is triggered when a submission is overdue (the students goes beyond the maximum submission date without submitting the work and the status of the thesis is still Pending). It notifies students and administrators of an overdue submission date.

OSS: Missing Academic Record Transcript Production - This workflow notifies the student when an administrator manually updates the details of a completion date of a missing academic records transcript. It is triggered when an administrator manually updates the details of a completion date of a missing academic records transcript.

OSS: Notify About Missing Academic Record - This workflow notifies the administrator about the missing academic records when a transcript request is placed by a student or by an administrator. It is triggered when a transcript request by a student or administrator indicates missing academic records.

OSS: Notify Admin About Transcript Production - This workflow notifies the administrator about the production of the transcript.

OSS: Notify Program Transfer Details - This workflow is triggered when a program transfer is committed (when the transfer itself is committed, whether or not the units or unit sections are transferred as part of the program transfer). It notifies students and administrators of program transfer changes. This workflow is not triggered if unit attempts or unit set attempts are transferred later in a separate transaction.

OSS: Notify Student About Intermission - This workflow is triggered when an intermission is committed or when an intermission that requires approval is committed or committed and approved. It notifies students and administrators of the intermission and intermission changes.

OSS: Notify Student About Transcript Production - This workflow notifies the individual student who placed an order about the production of the transcript. It is triggered when the system prints the academic records order placed by the student.

OSS: Notify Student for Program Discontinuation - This workflow is triggered when a program attempt is discontinued. The discontinuation can occur as the result of Discontinuation in Enrollment, as the result of transferring from a secondary program to a new primary program, or as the result of a progression outcome from progression. It notifies students and administrators of program discontinuation.

OSS: Notify Student Program Offering - This workflow is triggered when a program offering option is changed. This occurs when an administrator commits the change for a program offering option. For example, the administrator could use the Program Change button on the Student Enrollments window to change and commit the change of a program offering option. It notifies students and administrators of changes to program offering options.

OSS: Notify Students About Transcript Hold - This workflow notifies the student about a transcript hold when a transcript is requested by an administrator for a student who has a transcript hold. It is triggered when a transcript is requested by an administrator for a student who has a transcript hold.

OSS: Research Topic Modification - This workflow is triggered when a research topic is entered, changed, or approved. It notifies students and administrators of changes to research topics.

OSS: Thesis Topic Creation - This workflow is triggered when a thesis topic or title is entered, changed, or approved. This occurs when an administrator enters or changes the information for the thesis topic or title and saves the changes. It notifies students and administrators of changes to thesis topics.

Outcome Status - A workflow notification is sent to users alerting them to a change in the outcome status. This workflow is triggered when the outcome status is changed and saved.

Post Admission Requirements - This workflow allows administrative users to notify applicants of the additional documents or missing items that are required by the institution even after the admission application process is complete. This workflow is triggered by running the Send Notifications for Admission and Post Admission Requirements concurrent process.

Receive Error - This workflow is triggered if there is an error in the receipt of a reply from the degree audit trading partner.

Receive Reply - This workflow is triggered on receipt of a reply from the degree audit trading partner.

Residency Event - This workflow is triggered by the `oracle.apps.igs.pe.residency_change` business event. A workflow notification is sent to the system administrator by default.

Saved Transcript - This workflow is triggered by a business event.

Set or Release External Hold - Third party software or an external system raises an Oracle Student System designated business event in Oracle Workflow in order to insert or update a hold type into Oracle Student System. This workflow is triggered by third party software to apply or release an external hold.

Student Employment Notification - This workflow is triggered by the Student Employment Upload Payroll Information concurrent process and payroll details of the Student Work Award Progress window. It notifies users if the amount resulting from the subtraction of the amount of a student's work study award from the student's gross pay is within the threshold level.

Submit Request - This workflow is triggered by the Submit button on the Review Request self-service page to signal Oracle XML Gateway to begin processing the outbound request.

Thesis Exam - This workflow is triggered when the Thesis Exam business event is raised. This workflow notifies the student, the student's supervisors, and an administrator when the thesis has been submitted or resubmitted for the research candidacy.

Thesis Result - This workflow is triggered when the Thesis Result business event is raised. This workflow notifies the student, the student's supervisors, and an administrator of a change in the thesis result for the research candidacy.

Transcript Change Validation - When a new transcript is created in Admission, this workflow validates advanced standing records to check if the existing advanced standing still applies for the student. If not, then the advanced standing records are deleted or updated and a notification is sent to the staff member.

HRMS Applications

Oracle Federal Human Resources

GHR Personnel Action Process - Enables the routing of the Request for Personnel Action (RPA) Form for data entry, signature, and review before the final approval and update to the database. Based on the agency's practices, the user can route the RPA to an individual, groupbox, or routing list within the routing group. As the RPA is routed, the system maintains a history of actions. By referring to the history, users can learn what action was taken, by whom, and on what date.

GHR Position Description Process - Enables the routing of the Position Description form for data entry, signature, review and classification. Based on the agency's practices, the user can route the Position Description form to an individual, groupbox, or routing list within the routing group. As the Position Description form is routed, the system maintains a history of actions. By referring to the history, users can learn what action was taken, by whom, and on what date.

GHR Within Grade Increase Process - Enables the automatic processing of Within Grade Increase (WGI) actions without any manual intervention. The default WGI process automatically notifies the Personnel Office of the WGI approval and requires no response. WGI process can be configured during implementation in many ways based on the agency's practices.

Oracle Human Resources

Task Flow Item Type - Oracle Human Resources provides a predefined workflow item type called HR Task Flow that you can use to set up your task flows. The HR Task Flow item type includes a function activity for every HR application window that is allowed to be incorporated into a task flow. You can use these predefined function activities to model a workflow process for each task flow. Moreover, each function activity includes activity attributes that you can set to create button labels and position buttons on its corresponding application window.

The HR Task Flow item type provides you with an alternative to using forms to set up and maintain your task flows. By integrating with Oracle Workflow, you can use the graphical Oracle Workflow Builder to help you design and diagram the sequence of your windows.

Oracle iLearning

Oracle iLearning Workflow Process - Includes workflows for class enrollment, external learning, competency updates, and Order Management.

- Enroll in a Class
  - Checks to see if an existing class is full, then notifies learner of enrollment or placement on waiting list
  - If enrollment cancellation occurs too close to the class, cost transfer can take place, charging the customer for the enrollment
  - When Oracle iLearning is set to cross charge automatically, notifies class owner when it cannot find the Transfer From or Transfer To values; warns Administrator that they must manually create finance headers and lines
  - Notifies enrollment request creator of changes to enrollment status
  - Notifies learner and supervisor of enrollment cancellation

    - Notifies learner of enrollment cancellation caused by approver rejection
    - If a class is cancelled too close to its scheduled date, reminds class owner to move enrollees to waitlist manually
    - Notifies Learning Administrator of database errors that prevent automatic enrollment
    - Notifies System Administrator of database errors following enrollment approval
  - External Learning
    - Notifies a learner that the record of their attendance in a specific event has been recorded, updated, or deleted
  - Competency Update
    - Notifies learners and managers of automatic competency updates following successful class completion
    - Notifies managers of the need for manual competency updates
    - Manages competency update approvals and rejections
  - Learning administration through Order Management
    - Enables invoicing from the order line only after the student has completed the class
    - Notifies student of class or enrollment cancellation
    - Notifies class owner of enrollment or order line cancellation, and can enroll students from the waiting list
    - Reminds owner of cancelled class to manually delete booked resources
    - Notifies class owner when the maximum number of class attendees has increased
    - Notifies class owner when a customer has switched enrollments from one class to another

Oracle Internet Time

PA Timecard Approval Process - This process is initiated when an employee submits a timecard in Oracle Internet Time. The workflow can be configured to either automatically approve all timecards or route the timecard through a pre-determined approval process. The approval process sends notifications to managers and employees, ensures timecards adhere to company policy, and checks manager approval levels. The status of submitted timecards can be monitored throughout the approval process.

Oracle Self-Service Human Resources

Apply for a Job Process - You use this process to enable SSHR users to submit an application for a job. If the application is successful and the recruiter accepts the application, the supervisor of the vacancy receives a notification and the workflow completes.

Appraisals Processes - The appraisals workflows enable both managers and non-managers to use the appraisals functionality. Managers can create appraisals for their direct reports and generate notifications of the appraisal to the appraisee and any other appraisers. When a standard or 360-Degree appraisal has completed, the workflow can automatically trigger an update to the appraisee's competency profile. The workflow

also generates a Performance Review event. SSHR includes the following appraisals workflows:

- *Appraisal Details Process*
- *Appraisal Approved Process*
- *Appraisal Rejected Process*
- *Main Appraiser/Appraisee Notification Process*
- *Notify Participants Process*

Approvals Process - You can configure your SSHR workflow processes so that some transactions, for example, a change to an address, require approval whereas other transactions, for example, a change to a phone number, do not require approval. Related processes include:

- *Approval Initialisation*
- *Approval Notification Process*
- *Approved Process*
- *Commit Transaction Data Process*
- *Dynamic Approvals Process*
- *OnApproval Validation*
- *Rejected Process*

Approvals Process with Correction - In addition to the standard approvals functionality, this process also enables an approver to return an SSHR transaction to the initiator for correction of information or for additional information. The Approvals Process with Correction is the base process for approvals routing and notifications.

Change Extra Information Types/Change Special Information Types - These processes provide navigation for Extra Information and Special Information pages connecting to review, confirmation and approval processes. You can include the Extra information and Special information functions in other workflows to capture additional information if required.

Check Salary Basis Change in Mid-Pay Period - This process enables a SSHR manager to change the salary basis for a direct report during a pay period. The workflow generates a notification for the Payroll Contact who can then approve or reject the change.

Compensation Distribution - This process enables an SSHR manager to assign a one-time or recurring award to an employee or worker. Non-manager users can use this workflow to set up voluntary contributions.

Employee Review Process - This process enables a user to search for an existing review or create a new review. When the review is created, the workflow initiates the approval process.

Events and Bookings Processes - These processes generate notifications for users in the following situations:

- If a user has been added to an event (*Event & Bookings: Added to an Event*)
- If a user has been designated the internal contact for an event (*Event & Bookings: Notify Internal Contact*)
- If a user has been removed from an event (*Event & Bookings: Remove from an Event*)

Hiring Processes - The hiring processes control the sequence of events that occur when a recruiting manager hires a new employee or worker. The recruiting manager enters personal and assignment information for the new hire and then defines the pay and the new manager. The workflow processes are currently:

- *Hire or Placement*
- *French Hire*

HR Background Cleanup Process - This process removes workflow processes that are left running if a system crashes or if a user ID is disabled or removed.

Leave of Absence Processes - The Leave of Absence processes enable a user to enter, create, update, and confirm a leave of absence. The processes are:

- *Leave of Absence*
- *Leave of Absence Confirm*
- *Leave of Absence Create*
- *Leave of Absence Update*

Manage Employment Events Processes - SSHR includes several workflows to enable SSHR managers to perform Manage Employment Events transactions. For example, a manager can change the location, pay, or working conditions for an employee or contingent worker. The delivered workflows enable you to change the following information for an employee or contingent worker, either individually or as a combination:

- Assignment
- Cost Center
- Hours
- Job
- Location
- Manager
- Pay
- Role

The Manage Employment Events processes are built so that you can combine particular sets of pages and build a process with or without approval and with specific sequence of functions to control the page navigation.

New Employee Registration Processes - These processes combine to enable a new employee to register and enter personal and other information using SSHR. The new user can also create a new user name and password for SSHR. The processes include:

- *COBRA Registration Process*
- *Create User Name Process*
- *New Employee Registration Address Process*
- *New Employee Registration Assignment Process*
- *New Employee Registration Contacts Process*
- *New Employee Registration Process*

Notification Processes - These processes determine whether a user receives a notification on approval of a transaction or on submission of the transaction. The processes are:

- *OnApproval Notification Process*
- *OnSubmit Notification Process*

Organization Manager - The Organization Manager processes enable an SSHR manager to view or update organization managers.

Personal and Professional Information - SSHR includes workflows to enable users of SSHR (managers and non-managers) to enter, update, and delete personal and professional information. Users can currently enter, update, and delete information in the following areas:

- Personal Information
  - Address (secondary and main)
  - Basic Details
  - Contacts
  - Phone Numbers
- Professional Information
  - Academic Rank (US)
  - Competences
  - Education and Qualifications
  - Professional Awards
  - Tenure Status (US)

Process Payroll Payments Process - This process controls the Manage Payroll Payments function which an employee can use to determine how a salary is paid. If the user changes the payment method, SSHR calls the Change Payment Method Process.

Release Information for Transfer - This process enables a SSHR user to use the Release Employee Information function to share information about themselves with another user, often a manager, who would not usually have access to their records. Similarly, a manager can use this function to share information about one of their employees or workers with a second manager.

Return for Correction Process - This workflow process enables a SSHR user to return a transaction to the initiator for correction. This process is introduced in SSHR version 5 and controls notifications if they are returned for correction.

Self-Service HR Standard Error Process - This workflow process is triggered if a system error occurs. The workflow generates a notification to the SSHR Administrator to inform them of the system error.

Tax Processes

- *Online Tax Forms Notification Process* - This process sends a notification to an employee or other contact person if W4 tax records are changed.
- *W4 Information Processes (US)* - The Federal W4 Notification and Change W4 Information processes enable a US user to view W4 information and update the information if necessary. The standard setup for these processes does not require approval.

- *Employee W2 Process (US)* - This process enables a user to view tax information using SSHR. If the user requests a paper copy of the information, the workflow generates a notification for the HR or Payroll Representative to inform them of the request.

Training Enrollment Processes - The training enrollment processes enable users to search for training, enroll in a training offering, and cancel training. If appropriate, you can configure the processes to enable approvals by a manager or another person, for example, an HR administrator. The processes are:

- *Cancel Enrollment Process*
- *Complete Enrollment Process*
- *Enroll in Training*
- *External Training*
- *Training Link Access*

Oracle Time and Labor

OTL Seeded Approval Workflow - Oracle Time and Labor provides a single workflow that routes timecards to the appropriate person or people for approval. You use this workflow as a template to create as many workflows as you need for the approval styles you define.

Leasing

Oracle Lease Management

OKL CS Equipment Exchange - The workflow is used for Equipment Exchange Requests to update equipment details after receiving necessary approvals: 1. Notify Rejection of Equipment Exchange; 2. Notify to follow up with Customer about Temporary Exchange; 3. Notify modify contract for equipment exchange request. The workflow is initiated from the Requests tab.

OKL CS Transfer Assumption Request - The workflow is used for Transfer and Assumption Requests to notify the Contract Administrator after receiving necessary approvals: 1. Received Customer Approval? 2. Received Vendor Approval? 3. Notify Credit Departments. 4. Notify Collections for Approval. 5. Notify administrator to restructure contract. The workflow is initiated from the Requests tab.

OKL CS Billing Correction Request - The workflow is used for Billing Correction Requests to notify the Contract Administrator to approve billing correction requests. The workflow is initiated from the Transactions tab.

OKL CS Billing Refund Request - The workflow is used for Billing Refund Requests to notify the Contract Administrator to approve billing refund requests. The workflow is initiated from the Account tab.

OKL CS Convert Interest Type - The workflow is used to convert Interest Requests to notify the Contract Administrator of changes in the interest type from Lease Center. The workflow is initiated from the Structure tab.

OKL CS Contract Lease Renewal - The workflow is used for Renewal Quotes to notify the Contract Administrator to approve contract lease renewals: 1. Notification for Lease Renewal; 2. Notification for Lease Renewal Rejection. The workflow is initiated from the Requests tab.

OKL Stream Generation - Outbound - The workflow is used to initiate the outbound XML Pricing Engine transaction for Stream Generation: 1. Receive raised Business Event to send the outbound XML document; 2. Call XML Gateway API to check for validity of the generated XML document; 3. Report error, if applicable; 4. Call XML Gateway API to send the document to the Pricing Engine server. The workflow is initiated from the Stream Generation Process.

OKL Stream Generation - Inbound - The workflow is used to manage the inbound XML Pricing Engine transaction for Stream Generation: 1. Capture the Business Event for inbound XML message; 2. Process the streams returned by the Pricing Engine server. The workflow is a continuation of the Stream Generation Process.

OKL - INS Gather Third Party Insurance Information - The workflow creates a task to gather third-party information when a customer fails to provide insurance proof by the due date and the lessor cannot sell insurance. The workflow is initiated from the insurance placement process.

OKL Account Generator - You can use this workflow to generate account combinations instead of the seeded sources. The workflow is initiated from the Accounting Engine.

OKL - AM: Approve Contract Portfolio - The workflow routes a contract portfolio for approval. The workflow is initiated from Contract Portfolio.

OKL - AM: Notify Contract Portfolio Execution - The workflow notifies the Remarketer (Assignment Group) contract portfolio execution. The workflow is initiated from Contract Portfolio.

OKL - AM: Approve Restructure - The workflow routes a restructure for approval. The workflow is initiated from Restructure Quote.

OKL - AM: Shipping Instructions - The workflow is used to notify the lessee of shipping instructions for an asset. The workflow is initiated from Communicate Quote.

OKL - AM: Notify Internal Transport Department - The workflow is used to notify the internal transport department of the shipping details for an asset when the lessee accepts the transport management option. The workflow is initiated from Communicate Quote.

OKL - AM: Send Quote - The workflow is used to notify the requestor of an Asset Quote. The workflow is initiated from Communicate Quote.

OKL - AM: Repurchase Acceptance - The workflow informs users of the acceptance of a Repurchase Quote. The workflow is initiated from Quote Acceptance.

OKL - AM: Termination Quote Acceptance - The workflow informs users of the acceptance of a Termination Quote (Pre-Proceeds Termination Quote, Post-Proceeds Termination Quote). The workflow is initiated from Quote Acceptance.

OKL - AM: Restructure Quote Acceptance - The workflow informs users of the acceptance of a Restructure Quote. The workflow is initiated from Quote Acceptance.

OKL - AM: Notify Remarketer - The workflow informs the remarketing team of a new asset assignment. The workflow is initiated from Asset Return.

OKL - AM: Notify Collections - The workflow informs the collections agent of updates on returns in the case of a Repossessed Asset. The workflow is initiated from Asset Return.

OKL - AM: Notify Repossession Agent - The workflow informs the external repossession agent of a new repossession request. The workflow is initiated from Asset Return.

OKL - AM: Asset Repair - The workflow routes asset repair requests for approval. The workflow is initiated from Asset Condition.

OKL - AM: Request Title Return - The workflow requests the third party titleholder to return the title on expiration of the contract. The workflow is initiated from Remarketing.

OKL - AM: Remarketing Order Cycle - The workflow defines the order cycle in Order Management for remarketed assets. The workflow is initiated from Remarketing.

Credit Application Request - The workflow notifies the Lease Credit Approver to approve the recommendation. The workflow is initiated from Credit Management's recommendation business event.

Lease Funding Approval - The workflow is used to obtain approvals required for funding request approval. The workflow is initiated from the Funding Request submit button.

Lease Contract Booking Approval - The workflow is used to obtain approvals required for contract booking and activation. The workflow is initiated from the Approval button in Authoring.

OKL - AM: Notify Internal Transport Department - The workflow notifies the Transport Department when the lessee accepts the transport management option. The workflow is initiated from the shipping instruction.

OKL - AM: Notify Manual Termination Quote Request - The workflow notifies the representative of a manual termination quoting request. The workflow is initiated from the manual termination quote.

OKL - AM: AM Service Contract Integration - The workflow initiates Service Contract Integration. The workflow is initiated from the asset return.

1. Notify linked lease on asset return. The workflow is initiated from the asset disposal.
2. Notify linked lease on asset disposal. The workflow is initiated from the termination and billing.
3. Notify termination and subsequent contract billing. The workflow is initiated from the delinking at termination.
4. Notify delink and termination. The workflow is initiated from the termination.
5. Notify delink error and termination. The workflow is initiated from the Transaction tab.

OKL: CS Credit Memo - The workflow is used for getting approval before processing credit memos, with a notification for approving Credit Memo Request. The workflow is initiated from the Transaction tab.

OKL: Customer Service Principal Paydown - The workflow is used to send notifications about Principal Paydown, with a notification for Contract Rebook after paydown. The workflow is initiated from the Request tab.

OKL - SS: Request Termination Quote - Sends a notification to the Lease Center Agent that a quote has been requested. The workflow is initiated in Customer Self Service when a user creates a termination quote.

OKL - SS: Make a Payment - Sends a notification to the Lease Center Agent that a payment was made. The workflow is initiated in Customer Self Service when a user makes a payment.

OKL - SS: Request Repurchase Quote - The workflow creates a repurchase quote for the contract and sends out a notification for approval. The workflow is initiated in Customer Self Service when a user creates a repurchase quote.

OKL - SS: Cancel Insurance - Sends a notification to the Lease Center Agent that a policy has been cancelled. The workflow is initiated in Customer Self Service when a user cancels an insurance policy.

OKL - SS: Update Serial Number - The workflow sends out an approval notification and updates the serial number of the asset if approved, and sends a notification to requestor, whether the workflow is approved or declined. The workflow is initiated in Customer Self Service when a user requests a serial number update.

OKL - SS: Update Asset Location - The workflow sends out an approval notification and updates the location of the asset if approved, and sends a notification to requestor, whether the workflow is approved or declined. The workflow is initiated in Customer Self Service when a user requests an asset location update.

OKL - SS: Request Billing Change - The workflow sends out an approval notification and updates billing information of the contract if approved, and sends a notification to requestor, whether the workflow is approved or declined. The workflow is initiated in Customer Self Service when a user requests a billing change.

OKL - SS: Submit Third Party Insurance - If the insurance company does not already exist, the workflow sends out a notification to create a new party. The workflow creates an insurance policy once the notified user inputs the new party ID into the notification. The workflow is initiated in Customer Self Service when a user submits information on a new provider or when a user submits insurance policy details.

OKL - SS: Request For Invoice Format Change - The workflow sends out an approval notification and updates the invoice format if approved. The workflow sends a notification back to the user with the approval decision. The workflow is initiated in Customer Self Service when a user requests an invoice format change.

OKL - SS: Claim Notification - Sends a notification to the Lease Center Agent that a claim has been created. The workflow is initiated in Customer Self Service when a user submits a new insurance policy claim.

OKL - SS: Asset Return -The workflow notifies another user (the role selected in a setup step) to create a new asset return request. The workflow is initiated in Customer Self Service when a user submits a request for asset return.

OKL - SS: Request Renewal Quote - The workflow notifies the Lease Center Agent that a new renewal quote has been created. The workflow is initiated in Customer Self Service when a user creates a renewal quote.

OKL: Collections Approve Refund - The workflow is used to approve a vendor refund. The workflow is initiated from the Request Details.

OKL: Collections Approve Cure Request - The workflow is used to approve a Cure Request. The workflow is initiated when a user accepts a lease quote payment plan.

OKL - SO: Quote Acceptance - The workflow notifies another user that the quote payment plan has been accepted.

OKL - SO: Payment Plan Approval - The workflow routes the payment plan for approval. The workflow is initiated when the user submits a payment plan for approval.

Maintenance Applications

Oracle Enterprise Asset Management

EAM Work Request Approval Process - If this process is enabled within the Enterprise Asset Management parameters (Auto Approve check box), every user with a responsibility that is assigned to the work request's asset's current owning department will receive an electronic notification to take action on the work request created. These notifications can be viewed within eAM's Maintenance User, within the responsibility used to log in. Everyone receiving the notification can access the work request to change its status or add additional information to the work request's log. Once one user approves the work request, the notification will be removed from the Worklist of other users belonging to that same approval group. You can also reassign a notification to another user for approval or additional information.

If this process is enabled, work requests are created with a status of Awaiting Work Order (approved). Otherwise, work requests are created with a status of Open.

Manufacturing Applications

Oracle Cost Management

Account Generation Extension - The Account Generation Extension workflow is designed to provide a graphical user interface to the Client Extension. The workflow enables you to specify the accounts for distribution based on the type of transaction or to define your own business functions to generate the desired accounts. The Account Generation Extension workflow is called from the Account Generator Client Extension.

Oracle Engineering

Engineering Change Orders Process - Submits an engineering change order to the appropriate people for approval.

Oracle Inventory

INV: Move Order Approval - Sends notifications and reminders to approvers for approval of move orders, and sends notifications of the proposed move to requesters and people on the notification list associated with the subinventories.

Material Shortage Message - Sends a notification to planners every time a receipt transaction is done for an item with material shortages. The workflow notification will be initiated from a concurrent program that checks all material transactions that have been marked by the transaction manager.

Oracle Master Scheduling/MRP

Planning Exception Messages - Enables you to automatically process exceptions in order to take corrective action quickly when generating materials and resource plans. Exception messages are routed to key contacts for specific responsibilities. Recipients can be supplier and customer contacts as well as internal personnel.

Oracle Process Manufacturing (OPM)

Oracle Process Manufacturing components include the following workflows.

Using Oracle Order Management with Process Inventory

Sales Order Batch Reservations - The Sales Order Batch Reservations workflow sends notifications to the Customer Service Representative for changes in the production warehouse, quantity, completion date of the batch, and batch status.

OPM Product Development

Product Development Status Approval Workflow - When you activate the Product Development Status Approval Workflow, formulas, routings, recipes, validity rules, and operations require a series of approvals that result in reassigning their statuses through a predefined approval process. The workflow presents each approver with an electronic notification so action can be taken online. Laboratory Approval is optional in this workflow.

If the workflow is enabled, then formulas, routings, recipes, validity rules, and operations use the following statuses:

- Approved for Laboratory Use - The status changes to Request Approval for Laboratory Use until all approvers have accepted, at which time the status changes to Approved for Laboratory Use.
- Approval for General Use - The status changes to Request Approval for General Use until all approvers have accepted, at which time the status changes to Approved for General Use.

If the status is Approved for Laboratory Use, then you can only use formulas, routings, recipes, validity rules, and operations in laboratory batches and cost rollups for laboratories.

If the status is Approved for General Use, then you can use formulas, routings, recipes, validity rules, and operations in production batches and cost rollups.

OPM Quality Management

Sample Creation Notification - Sends a notification to draw a physical sample as required by the associated sampling plan for the specification and specification validity rule in effect. This workflow can be triggered by an inventory quantity increase, receiving transaction, batch step release, or lot expiration or retest. The notification lists the sampling plan information and includes a link to the Samples window, where information about the sampling event is populated.

Testing Notification - Sends a notification to prompt you to perform a test and enter its result against a particular sample. This workflow is initiated by the creation of a sample that is already associated to a specification or selection of the result action for retesting a certain test. Each tester for a given sample in process receives a notification for the expected test result.

Sample Disposition Notification - Prompts for assignment of the final disposition for a sample once testing has completed. The sample disposition can be changed to accepted, accepted with variance, or rejected when all tests (if required by the associated specification) have recorded results that have been evaluated and finalized.

Composite Results Notification - Asks you whether to composite the results across a set of samples within the same sampling event. This workflow is initiated when testing of all the samples in a sampling event (based on the number of samples required by the sampling plan) is complete and each sample has a final sample disposition. The notification provides a link to the Composite Results window, where information about the sampling event is populated.

Sample Group Disposition Notification - Notifies you that the sampling event requires a final disposition, based on the number of samples required by the sampling plan. This workflow is launched when all the samples within a sampling event have a final disposition of accept, accept with variance, reject, or cancel.

Spec Status Change Approval Notification - The Spec Status Change Approval Notification workflow notifies the contact person when the specification status has changed. The workflow is activated for processing the status change. Typically, workflow approval is activated for status changes to Approved for Lab Use and Approved for General Use. The other statuses can be processed without requiring workflow approval.

Spec Customer Validity Rule Status Change Approval - The Spec Customer Validity Rule Status Change Approval Notification workflow notifies the contact person when the customer specification validity rule status has changed. The workflow is activated for processing the status change. Typically, workflow approval is activated for status changes to Approved for Lab Use and Approved for General Use. The other statuses can be processed without requiring workflow approval.

Spec Inventory Validity Rule Status Change Approval - The Spec Inventory Validity Rule Status Change Approval Notification workflow notifies the contact person when the inventory specification validity rule status has changed. The workflow is activated for processing the status change. Typically, workflow approval is activated for status changes to Approved for Lab Use and Approved for General Use. The other statuses can be processed without requiring workflow approval.

Spec Supplier Validity Rule Status Change Approval - The Spec Supplier Validity Rule Status Change Approval Notification workflow notifies the contact person when the supplier specification validity rule status has changed. The workflow is activated for processing the status change. Typically, workflow approval is activated for status changes to Approved for Lab Use and Approved for General Use. The other statuses can be processed without requiring workflow approval.

Spec WIP Validity Rule Status Change Approval - The Spec WIP Validity Rule Status Change Approval Notification workflow notifies the contact person when the WIP specification validity rule status has changed. The workflow is activated for processing the status change. Typically, workflow approval is activated for status changes to Approved for Lab Use and Approved for General Use. The other statuses can be processed without requiring workflow approval.

Spec Monitoring Validity Rule Status Change Approval - The Spec Monitoring Validity Rule Status Change Approval Notification workflow notifies the contact person when the monitoring specification validity rule status has changed. The workflow is activated for processing the status change. Typically, workflow approval is activated for status changes to Approved for Lab Use and Approved for General Use. The other statuses can be processed without requiring workflow approval.

Sample Group Rejection Notification - The Sample Group Rejection Notification workflow notifies the contact person that either a monitoring sample group or a stability study time point sample group is rejected.

Stability Study Change Status Notification - The Stability Study Change Status Notification workflow notifies the study owner of status changes in the stability study. The process of changing status from New to a status of Completed involves a series of workflow notifications. Oracle E-Records operates with this workflow to enforce the stability study status control business rules. The cancellation of a stability study is also controlled by the workflow.

Stability Study Lot/Sublot Sample Notification - When the stability study is approved, and specific lots or sublots of the item to be tested exist, then the Stability Study Lot/Sublot Sample Notification workflow notifies the stability study contact person to obtain samples for all required material sources. In the notification, the contact person is designated to draw and store the specified samples, and to update the sample records accordingly.

Stability Study Batch Creation Notification - When the stability study is approved, and specific lots or sublots of the item to be tested do not exist, then the Stability Study Batch Creation Notification workflow notifies the stability study contact person to request that a batch be created for the study.

Stability Study Testing - The Stability Study Testing workflow is a master workflow that governs the Stability Study Time Point Scheduling Notification workflow, and the Stability Study Late Time Point Scheduling Notification workflow. This workflow runs as a background process to monitor for any time point test that requires a workflow notification, or a late workflow notification. The workflow raises the appropriate event for either scheduled or late time point tests.

If the stability study is canceled or the scheduled time point testing date changes, then a workflow API manages the situation. The sample group disposition is changed from a status of Retained to a status of Pending, and the sample pull dates are updated.

Stability Study Time Point Scheduling Notification - The Stability Study Time Point Scheduling Notification workflow notifies you to pull samples for time point testing. The notification considers the required leadtime, and it requests you to change the disposition of a time point sample from Retained to Pending. When the sample disposition is changed to Pending, tests are scheduled. If the test is not performed within the grace period, then the OPM Quality Stability Study Testing master workflow raises a separate event that indicates the test is overdue.

Stability Study Late Time Point Scheduling Notification - The Stability Study Late Time Point Scheduling Notification workflow notifies you that a test is overdue. The notification considers the required grace period, and uses the OPM Quality Stability Study Testing master workflow to raise a separate event to indicate the test is late.

Quality UOM Conversion Notification - The Quality UOM Conversion Notification workflow notifies the contact person that a new or revised lot-specific UOM conversion is recommended based on quality results. A link to the Item Lot/Sublot Standard Conversion window in the OPM Inventory Control application is provided. This link does not operate unless a lot number is assigned on the quality sample.

OPM Inventory

Lot Expiry/Retest Workflow - Sends a notification to a user at a defined number of days in advance of lot expiration and retest dates. Without this workflow the expiration or retesting of a lot is not visible to the user without a query for the expiration and retest information. The Lot Expiry/Retest Workflow gives the approver this visibility by notifying the approver the defined number of days in advance of the expiration and retest date.

Item Activation E-Record - Sends notifications to approve newly created items before changing their status to an approved inventory status. The workflow is initiated from the Approve Item from the OPM Item Master window. The workflow notifies the individual who needs to approve the item's creation and calls for e-signatures using Oracle E-Records. The item becomes active if it is approved. The item remains inactive

if it is not approved. Item properties cannot be modified if the workflow approval is pending for an item.

Oracle E-Records

Oracle E-Records Individual Approval Notification - For all deferred e-signatures, the requestor receives a notification after each initiated e-signature of the event.

Oracle E-Records E-Signature Reminder Notification - Signers can receive reminder notifications that there are e-records awaiting their e-signatures.

Oracle E-Records Timeout Notification - An e-record requestor is notified when an e-record event times out when an e-signature is not received in the allotted period of time.

Oracle E-Records Rejection Notification - All signers and the requestor receive notifications when a rejection signature is entered for an associated e-record event.

Oracle E-Records Stylesheet Upload Notification - When all approvals have been captured for a new or updated E-Record style sheet file, the style sheet is automatically uploaded to the style sheet repository. A notification is sent to the requestor.

Oracle E-Records File Approval - Uploaded files can be routed for approval and e-signatures. This event is initiated from the Oracle E-Records File Upload window by clicking the Send for Approval button.

Oracle E-Records Transaction Configuration Variable Create/Update - The settings for Oracle E-Record transaction defaults are defined and set here. Enabling these events requires that e-Records and e-Signatures are captured before updates are saved. The new or updated data is not committed to the database unless all of the required signatures are obtained while the E-Signature window displays.

Oracle E-Records Rule Configuration Variable Create/Update - The settings for Oracle E-Record transaction rules are defined and set here. Enabling these events requires that e-records and e-signatures are captured before updates are saved. The new or updated data is not committed to the database unless all of the required signatures are obtained while the E-Signature window displays.

Oracle Project Manufacturing

Indirect/Capital Project Definition Process - This process is part of the Project Manufacturing Project Definition process navigator flow. This process guides users through all the necessary sequence of steps for setting up an Indirect- or Capital-type project for use in Oracle Project Manufacturing.

Contract Project Definition Process - This process is part of the Project Manufacturing Project Definition process navigator flow. This process guides users through all the necessary sequence of steps for setting up a Contract-type project for use in Oracle Project Manufacturing.

Oracle Quality Online

Employee Notifications - Notifies employees that have registered interest for a particiular event on a designated Defect or Enhancement.

Critical Defect - Notifies the owner that they have recently been assigned a critical defect.

Oracle Warehouse Management

MTL Transaction Reasons Workflow - Transaction reason actions provide the user with an alternative customizable function flow when an exception happens. This workflow currently has two workflow processes seeded:

- Cycle Count - Requests a cycle count on a location with a quantity discrepancy and then send out a notification.
- WMS N Step Putaway - Allows the user to putaway items to an intermediate location that is different from the location suggested by the system. The system then creates another task to move the item from the intermediate location to the suggested location.

Oracle Work in Process

Intermediate Shipment - Enables tracking the outside processing assemblies from the shop floor to the shipping dock, and then to the supplier. The Intermediate Shipment workflow is activated when you move assemblies into the Queue intraoperation step of an Outside Processing operation with a PO Move resource charge type, or if the outside processing operation is the first operation.

Order Fulfillment

Oracle Configure To Order

CTO Change Order - Sends a notification to the planner in the shipping organization detailing the changes made to an ATO order. It is started when a change to the order quantity, schedule ship date, request date, or schedule arrival date is made and a discrete job reservation or a flow schedule exists for the order. It is also started anytime a configuration change or order line cancellation is made, even if a reservation does not exist.

Oracle Order Management

Order and Line Runnable Processes and Functional Subprocesses - Oracle Order Management provides seeded order and line runnable processes and functional subprocesses. Order Header workflow data is defined under the item type OM Order Header (OEOH). Seeded header runnable processes are provided to support the processing of standard Orders and Returns with approvals. Booking and Close Order functional subprocesses are also seeded. Order Line workflow data is defined under the item type OM Order Line (OEOL). Oracle Order Management comes seeded with line-level runnable processes to support the processing of standard items, configurations, service items, drop-shipments, returns for credit only, and returns for credit with receipt and approval. Functional subprocesses to schedule, ship, fulfill, invoice interface, and close order lines are also seeded. Additionally, you can configure custom processes to meet your specific business requirements using the seeded functional subprocesses and custom activities. You can use these seeded and custom runnable processes for order processing by assigning them to specific order and line transaction types. Oracle Order Management usually seeds new processes with every release. Please refer to the *Order Management Workflow User Guide* for the latest Oracle Order Management processes.

Change Order Notification Process - Sends a change order notification from certain application forms. The recipient and message content are set dynamically when you select a responsibility and provide content. Uses the item type OM Change Order (OECHORD).

COGS Process - Generates a cost of goods sold account using the item type Generate Cost of Goods Sold Account (OECOGS).

Negotiation Header Processes - Oracle Order Management supports the negotation phase of a sales order (that is, quotes) or blanket sales agreement. The item type for Negotiation is OM Negotiation Header (OENH). It includes steps such as submit draft, optional internal approval, customer acceptance, and complete negotation. If the transaction is a quote, at the end of the negotiation flow, the OM Order Header flow will be started; while if the transaction is a blanket sales agreement, at the end of the negotiation flow, the OM Blanket Header flow will be started.

Blanket Header Process - Oracle Order Management supports Blanket Sales Agreements through the workflow item type OM Blanket Header (OEBH). It includes steps such as submit draft (if the Blanket has no negotiation phase), expiration date handling and reminders, terminate blanket, and close blanket.

Order Import Process - Oracle Order Management supports the creation and modification of sales orders through the Order Import workflow. When an inbound XML message is received, the Order Import workflow is triggered and, depending on the setup, order import is either run synchronously or deferred for an asynchronous run. This workflow process subscribes to the event `oracle.apps.ont.oi.po_inbound.create` and its item type is OM Order Import (OEOI). It includes steps such as raising an event to trigger the OAG CONFIRM BOD XML message, interface table data population, raising integration events, and conditional execution of order import.

Send Acknowledgment Process - Oracle Order Management supports the generation of the ACKNOWLEDGE PO and CONFIRM BOD XML message. Both messages conform to the Open Applications Group (OAG) standard. The CONFIRM BOD notifies the buyer that an inbound document has been received and the ACKNOWLEDGE PO conveys to the buyer the results of order import. The CONFIRM BOD and ACKNOWLEDGE PO utilize different processes in the same workflow. These processes subscribe to the events `oracle.apps.ont.oi.cbod_out.confirm` and `oracle.apps.ont.oi.po_ack.create`, respectively. The item type for the workflow is OM Send Acknowledgment (OEOA). The two processes include steps such as raising integration events, generation of the deliverable data, and document transmission.

Show Sales Order Process - Oracle Order Management supports the generation of the SHOW SO and CHANGE SO XML messages. Both messages conform to the Open Applications Group (OAG) standard. SHOW SO contains status information concerning the current state of an order and CHANGE SO is used to notify the buyer of changes to his or her order. Both messages utilize the same process in the Show Sales Order workflow. The workflow process subscribes to the event `oracle.apps.ont.oi.show_so.create` and the item type is OM Show Sales Order (OESO). The value of a particular parameter in the consumed event determines which message will be generated. The flow includes steps such as checking trading partner setup, raising integration events, generation of the deliverable data, and document transmission.

Open Interface Tracking Process - Oracle Order Management supports the tracking of XML, EDI, and generic order import transactions through the Open Interface Tracking workflow. This workflow process subscribes to the integration event `oracle.apps.ont.oi.xml_int.status` and retrieves information from those events for use in the Open Interface Tracking UI. The item type for this workflow is OM Open Interface Tracking (OEEM).

EDI Workflow Process - Oracle Order Management supports the derivation and generation of EDI acknowledgment data through the OM EDI Workflow (OEXWFEDI) item type. This workflow process subscribes to the event `oracle.apps.ont.oi.edi_ack_values.create` and populates Oracle Order Management's acknowledgment tables with the data necessary to generate the outbound EDI messages (POAO and POCAO).

Procurement

Oracle iProcurement

For a complete list of Oracle iProcurement workflows, see the *Oracle iProcurement Implementation Guide*.

Receipt Confirmation Process - Sends receipt notifications to requestors, informing them that they should have received their order. This process is also known as the PO Confirm Receipt workflow.

Requisition Approval Process - Submits a requisition created from Web Requisitions to the appropriate managers for approval and updates the status of the requisition.

Oracle iSupplier Portal

For a complete list of Oracle iSupplier Portal workflows, see the *Oracle iSupplier Portal Implementation Guide*.

Supplier Self-Service Registration Approval Process - Oracle iSupplier Portal allows a guest to log on and register as a supplier contact for a company. This process routes a notification to the appropriate account approver to verify and approve the registration. If the approver approves the registration, the Supplier Web User account is activated. If the account approver rejects the registration, the account is deactivated.

Oracle Purchasing

For a complete list of Oracle Purchasing workflows, see the *Oracle Purchasing User's Guide*.

Procurement Workflow - The Procurement Workflow is a lights-out, hands-off transaction processing system that is truly flexible and extensible to all members of your supply chain. It is one of the key enablers in the shift towards more strategic sourcing and procurement activities. It consists of the Document Approval, Automatic Document Creation, Change Orders, Account Generation, Send Notifications, Price/Sales Catalog Notification, and Receipt Confirmation (used only by Self-Service Purchasing) workflow processes.

Document Approval Process - Performs all approval related activities in Oracle Purchasing. These include, but are not limited to, document submission, approval, forwarding, approval notifications, and rejection. This includes the PO Approval workflow process for approving purchase orders and the PO Requisition Approval workflow process for approving requisitions.

Automatic Document Creation Process - Automatically creates standard purchase orders or releases against blanket agreements using approved purchase requisition lines, if the requisition lines have the required sourcing information. This process is also known as the PO Create Documents workflow.

Change Orders Process - Allows you to control which changes require a manual reapproval and which will be automatically reapproved. All reapproved documents, either manual

or automatic, will result in the document revision being incremented. This process is part of the PO Approval workflow.

Send Notifications Process - Looks for documents that are incomplete, rejected, or in need of reapproval, and sends notifications regarding the document's status to the appropriate people. This is also known as the PO Send Notifications for Purchasing Documents workflow.

Price/Sales Catalog Notification Process - Sends a notification to the buyer when the price/sales catalog information sent through the Purchasing Documents Open Interface includes price increases that exceed a price tolerance that you set. This process is also known as the PO Catalog Price Tolerance Exceeded Notifications workflow.

Projects

Oracle Project Foundation

Project Forecasting Workflow - Generates the forecast for a project.

Project Approval and Status Change Process - Routes a project and notifies appropriate users of any project status change. For example, you can submit the project for approval, or notify appropriate people upon project closure. You select which workflow to use for the appropriate status change, as well as determining the person(s) to route the project to.

Oracle Project Costing

Step Down Allocations Process - Automates the execution of step-down auto allocation sets to create allocation runs, generate the allocation transactions, release the allocation transactions (or require approval before the process proceeds), distribute costs, and update project summary amounts.

Oracle Project Resource Management

Advertisements Workflow - Sends advertisement notifications and e-mails.

Apply Team Template - Handles the task of applying a team template to a project.

Assignment Approval Workflow - This process manages approvals for resource assignments. It is launched when an assignment is created for a resource and is submitted for approval. When the assignment is approved or rejected, FYI notifications are sent to the resource, the resource manager, the staffing manager, and the project manager.

Candidate Notification Process - This notification process is launched when a candidate is nominated or withdrawn for a requirement. The process notifies the resource, the manager of the resource, and their staffing manager of the nomination.

HR Related Updates Workflow - Synchronizes Oracle HRMS data with Project Resource Management data.

Mass Assignment Approval - Handles the routing of approvals for mass assignments.

Mass Assignment Transaction Workflow - Handles the creation of assignments when a mass assignment request is submitted.

Overcommitment Notification Process - Notifies users when assignments cause overcommitments.

Oracle Project Management

Budget Approval Process - Routes a project budget for approval and baseline. You select which workflow to use for the budget type, as well as determining the person(s) to route the budget to.

Budget Integration Workflow - When a project is set up to use bottom-up integration, the project budget baseline process launches this deferred workflow. This workflow process performs the following tasks:

- Validates the submitted budget version
- Optionally, activates the Budget Status Change workflow
- Interfaces the budget amounts for baseline versions to Oracle General Ledger

Project Budget Account Workflow - When a project budget is integrated with a General Ledger budget, an account must be assigned to each project budget line. The account is used to interface the project budget amount to General Ledger. The Project Budget Account workflow enables you to automate the account generation and assignment process.

Sales, Marketing, and eCommerce

Oracle Content Manager

Content Item Approval - Sends notifications to designated approvers when a content item is submitted for approval.

Translation Approval Workflow - Sends notifications to designated approvers when a content item is waiting for translation approval. After the content item is approved, translators are notified, so that they can translate the content item into different languages.

Oracle Incentive Compensation

Contract Approval Process - Oracle Incentive Compensation uses the Contract Approval workflow in the Sales Force Planning module to process an agreement for approval by management, distribute the agreement to the sales force, and allow the salespeople to accept the agreement. The Contract Approval process begins when the user submits an agreement for approval. After all approvals are done and the salesperson accepts the agreement, it is activated as a compensation plan. The Contract Approval Process sends messages to remind the manager to distribute the agreement, remind the sales force to accept the agreement, and alert the sales manager that an agreement has not yet been accepted.

Oracle iStore

Oracle iStore provides seeded workflows in which there are predefined notifications and notification messages. The iStore Alert Workflow contains the non-report-related notifications, while the report-related notifications belong to iStore Reports Alerts. The notification e-mail messages are sent to users based on various events.

Notifications Related to Orders

Order Confirmation Normal - Confirms a normal order. This notification is initiated when a user places an order in a Web specialty store.

Order Confirmation - Next Steps for Faxed Orders - Explains the remaining steps for order submission. This notification is initiated when a user places an order and chooses to fax a credit card or purchase order as payment.

Orders Not Booked Notification - Announces that an order has not been booked to the system administrator. This notification is initiated when a user's order is not booked.

Cancel Order - Confirms a cancelled order. This notification is initiated when a user cancels an order using the Web specialty store.

Notifications Related to Shared Shopping Carts

Share Cart Confirmation Notification - Confirms a shared cart event and explains how to access the shared cart to the Shared cart owner and recipients. This notification is initiated when a user shares a cart with other users.

Change Access Level - Notifies users about access level changes. This notification is initiated when a Shared cart owner changes the access level of the recipient.

Remove Cart Access for Recipient - Notifies users about the shared cart status. This notification is initiated when a Shared cart recipient ends working on a shared cart.

Stop Sharing - Notifies users that a shared cart has been un-shared by the owner. This notification is initiated when a Shared cart owner stops sharing a shared cart.

Notifications Related to Sales Assistance

Sales Assistance Request - To Sales Representatives - Describes a request for sales assistance to a sales agent. This notification is initiated when a user requests sales assistance during the checkout flow.

Sales Assistance Request - Sends a confirmation to users who requested sales assistance. This notification is initiated when user requests sales assistance during checkout flow.

Notifications Related to Users and Registration

User Registration - Welcomes a newly registered user. This notification is initiated when a user registers in a specialty store, or a B2B user is registered by the B2B Primary User.

Forget Login - Sends email to a user with his or her username and password. This notification is initiated when a registered user requests login information.

Notification Events Related to Contracts

Contract Negotiations Request - Disapproval - Announces rejection of a contract terms change request. This notification is initiated when the Contract administrator rejects a user's request for changes in contract terms.

Contract Negotiations Request - To Sales Representatives - Describes a request for changes to contract terms. This notification is initiated when a user requests changes in contract terms.

Contract Negotiations Request - Approval - Announces approval of a contract terms change request. This notification is initiated when the Contract administrator approves a user's request for changes in contract terms.

Contract Negotiations Request - Cancellation - Announces cancellation of contract negotiations. This notification is initiated when the Contract administrator cancels a contract that was created when a user requested changes in contract terms.

Contract Negotiations Request - To Users - Acknowledges a request for changes to contract terms. This notification is initiated when a user requests changes in contract terms.

Notification Events Related to Reports

Reports - iStore Historical Summary - Contains the information from the Store Order Summary Data Out Bin specified in the Store Administration UI. This notification is initiated when iStore Reports concurrent programs refresh the report data.

Reports - iStore Top Orders - Contains daily Top Orders from the Top Orders Bin specified in the Store Administration UI. This notification is initiated when iStore Reports concurrent programs refresh the report data.

Oracle Marketing Online

Marketing Approval Workflow - This process gets initiated when a user submits a marketing activity like a Campaign or Event for approval to make it Active. The Marketing Approval Workflow is used for concept approval, budget approval, and approval for any funds that may be required for running various marketing activities.

Marketing Generic Approval Workflow - This process gets initiated when a user submits Root Budgets, Claims, and Price Lists for approval to make them Active or Complete.

Schedule Execution - This process gets initiated when a campaign activity is set to Available or Active, which means that it needs to be executed. This could happen as a part of schedule's manual activation by the user, or as a part of any automated schedule execution activity such as triggers or repeating schedules. This process updates the schedule's status to Active if needed, and then performs any seeded execution functionality based on the schedule's channel.

Marketing Triggers - This process gets initiated when a Marketing trigger's next run time has been reached. This could be either the original start time of the trigger, or a scheduled time for the trigger repetition, if the trigger repeats at a specific interval. This process verifies, for non date based triggers, if the condition for the trigger is satisfied. If it is, then some actions need to be performed, which is done by some of the subflows. The types of actions that can be performed are execution of associated schedules, generation and execution of associated schedules that need to repeat through this trigger, sending out notifications, and plugged-in custom logic. Also, if the trigger repeats, then the next run date/time for it is calculated and scheduled.

Oracle Workflow Business Event System Implementation in Oracle E-Business Suite

Several Oracle E-Business Suite products leverage the Oracle Workflow Business Event System for business process integration. For a complete list of business events provided by Oracle E-Business Suite, please see the Oracle E-Business Suite Electronic Technical Reference Manual (eTRM), available on Oracle*MetaLink*. A description of each feature is documented in its respective product's User's Guide or Implementation Guide, if one is available.

Related Topics

Oracle Workflow Support Policy, page C-42

Oracle Workflow Implementation in the Oracle Technology Stack

The products listed below leverage Oracle Workflow for business process definition and integration. A full description of each feature is documented in its respective product's User's Guide or Configuration Guide, if one is available.

Oracle Warehouse Builder

Oracle Warehouse Builder includes a Workflow Deployment Wizard that lets you deploy extract, transform, and load mappings to Oracle Workflow as functions within an item type. You can then use Oracle Workflow Builder to define the sequence of these functions as a workflow process. In designing the process, you can specify job dependencies between the mappings to ensure that jobs run in the proper order. You can then run the process from Oracle Workflow or schedule the process to run using Oracle Enterprise Manager.

By defining jobs as a workflow process, you can automate the entire schedule for a set of jobs. When you run the process, Oracle Workflow manages the jobs so that they run in the sequence defined in the process. If an exception occurs, Oracle Workflow terminates the process. This approach minimizes the manual intervention required to manage warehouse load and refresh jobs.

For more information, see: Managing Dependencies Using Oracle Workflow, *Oracle Warehouse Builder User's Guide*.

Oracle Application Server InterConnect

The Oracle Workflow Business Event System enables Oracle Application Server InterConnect and Oracle Workflow to work together to provide a complete business process driven integration solution. With Oracle Application Server InterConnect and Oracle Workflow, you can define business collaborations across two or more applications to implement the business processes for an organization.

Simple business process definitions can be implicitly captured in the messaging defined through Oracle Application Server InterConnect core functionality. For more complex business processes, Oracle Application Server InterConnect leverages the robust design time and runtime Oracle Workflow business process definition and execution support to make the processes explicit and manageable. For example, Oracle Workflow allows you to model error management for exceptions, human interaction such as approvals, message junctions including both message fan-in and fan-out, stateful routing, and composite services involving communication across several applications.

The Oracle Application Server InterConnect iStudio design tool automatically generates Oracle Workflow business event and subscription definitions corresponding to common view events and procedures. You can launch the Oracle Workflow home page from iStudio to review these definitions.

The iStudio tool also deploys process bundles as Oracle Workflow item type definitions. These item types include starter workflow processes with Oracle Workflow event activities that correspond to Publish, Subscribe, Invoke, and Implement activities defined in iStudio. You can then launch Oracle Workflow Builder from iStudio to complete the workflow process definition by specifying the sequence of the event activities and optionally adding other activities such as notifications or functions.

For example, iStudio might generate a workflow process with two event activities, one that receives a CreatePO event and another that sends an AcceptPO event. You can then

use Oracle Workflow Builder to define the business process that controls the execution of these activities. For instance, add a notification activity to send an e-mail requesting approval after the CreatePO event is received and before the AcceptPO is event is sent.

At runtime, Oracle Application Server InterConnect and Oracle Workflow communicate with each other through the Oracle Workflow Business Event System, leveraging the Oracle Advanced Queuing messaging infrastructure, to execute business processes defined across multiple applications.

For more information, see: Oracle Application Server InterConnect and Oracle Workflow, *Oracle Application Server InterConnect User's Guide*.

Oracle Application Server Wireless

Oracle Workflow and Oracle Application Server Wireless are integrated to let you send wireless notifications. Oracle Application Server Wireless integrates with Oracle Workflow by providing a subscriber to the WF_NOTIFICATION_OUT queue. This subscriber dequeues notification messages from the queue as JMS Text messages and can then send them to wireless devices. If a user sends a response from a wireless device, Oracle Application Server Wireless calls the appropriate notification response function to record the response and complete the notification. For more information, please refer to the *Oracle Application Server Wireless Administrator's Guide* and the *Oracle Application Server Wireless Developer's Guide*.

Related Topics

Oracle Workflow Support Policy, page C-42

Oracle Workflow Support Policy

Oracle Workflow is embedded in Oracle Applications and is used by its modules to automate and streamline business processes. You can use Oracle Workflow Builder to easily modify an existing business process without changing its application's code. Oracle Workflow also allows you to extend your workflow processes as your business rules change and mature. Additionally, you can use the Event Manager to modify event and subscription definitions without changing application code.

Before you use Oracle Workflow to customize any predefined workflow process, event, or subscription, you should familiarize yourself with the following customization guidelines to ensure standard and safe design and development practices. By following these guidelines, you will be able to supply important information to Oracle Support Services in helping you resolve any issues that arise from your customizations.

Customization Guidelines

1. Verify that all setup steps have been completed as documented in the Oracle Workflow documentation, and the product-specific User's Guides.
2. Test the unmodified seeded workflow, event, or subscription on a test database and ensure that it runs successfully with the setup and data specific to your environment.
3. Refer to the product-specific User's Guide and any documentation update, available on Oracle*MetaLink*, for the specific workflow, event, or subscription of interest. These documentation sources specifically mention what should NOT be modified. Oracle

Support Services will not support modifications to any object that is specifically documented as not modifiable.

4. Gradually build in customizations step-by-step, and test the customized workflow or subscription after each step.
5. When creating PL/SQL procedures, conform to the standard PL/SQL API templates documented in the *Oracle Workflow Developer's Guide*. Be sure to handle exceptions in the event of an error so you can track down the procedure where the error has occurred.
6. Do not implement the customized workflow, event, or subscription in production without fully ensuring that it works successfully on a test database, which is a replica of your production setup.

Resolving Customization Issues

If you encounter a problem when customizing a seeded workflow, event, or subscription, you should:

- Provide the Support analyst with the modified workflow definition file or event or subscription definition, and where possible, identify the exact step where the problem occurred.
- Provide the Support analyst with results of running the unmodified seeded workflow or subscription.

What Is NOT Supported

The following types of customizations are not supported:

1. Modifying a workflow object that has a protection level that is less than 100.
2. Altering a workflow object's protection level if its original protection level is less than 100.
3. Modifying your access level to an unauthorized level of less than 100 for the purpose of modifying workflow objects that are protected at levels less than 100.
4. Customizations that are explicitly documented as being UNSUPPORTED in the seeded workflow's product-specific User's Guide or documentation update notes. This includes modifying processes, attributes, function activities, event activities, notifications, lookup types, messages, events, or subscriptions that are specifically documented as not to be modified.
5. Manual modification of Workflow tables with a prefix of `WF_` or `FND_` unless it is documented in the Oracle Workflow documentation or is required by Oracle Support Services.
6. Customizations to any predefined Oracle Workflow directory service view definitions, including any of the predefined implementations of the WF_USERS, WF_ROLES, and WF_USER_ROLES views, as well as the bulk synchronization views provided by originating systems within Oracle Applications.
7. Customizations to any predefined versions of the Oracle Workflow PL/SQL security package, WFA_SEC.

What Is Supported

The following types of customizations are supported:

1. Any customization that is stated as Required in the seeded workflow's, event's, or subscription's product-specific User's Guide or documentation update notes.
2. Customization examples documented in the product-specific User's Guide or documentation update notes. Any issues that arise are fully supported to resolution, to the extent that the customization example was followed as documented. Any deviation from what is documented amounts to a custom development issue that needs further evaluation. See number 3 below.
3. Customizations that are not explicitly stated as unsupported customizations, as required customizations, or as supported customization examples are supported to the extent that the customer must first isolate the problem following the customization guidelines discussed earlier. If upon evaluation, Oracle Support Services deems that the isolated problem stems from an Oracle product, Oracle will supply a solution. Otherwise, it is the responsibility of the customer to correct the custom development issue.

Glossary

Access Level

A numeric value ranging from 0 to 1000. Every workflow user operates at a specific access level. The access level defines whether the user can modify certain workflow data. You can only modify data that is protected at a level equal to or higher than your access level.

Activity

A unit of work performed during a business process.

Activity Attribute

A parameter that has been externalized for a function activity that controls how the function activity operates. You define an activity attribute by displaying the activity's Attributes properties page in the Activities window. You assign a value to an activity attribute by displaying the activity node's Attribute Values properties page in the Process window.

Agent

A named point of communication within a system.

Agent Listener

A type of service component that processes event messages on inbound agents.

Attribute

See Activity Attribute, Item Type Attribute, or Message Attribute.

Background Engines

A supplemental Workflow Engine that processes deferred or timed out activities or stuck processes.

Business Event

See Event.

Cost

A relative value that you can assign to a function or notification activity to inform the Workflow Engine how much processing is required to complete the activity. Assign a higher cost to longer running, complex activities. The Workflow Engine can be set to operate with a threshold cost. Any activity with a cost above the Workflow Engine threshold cost gets set to '`DEFERRED`' and is not processed. A background engine can be set up to poll for and process deferred activities.

Concurrent Manager

An Oracle Applications component that manages the queuing of requests and the operation of concurrent programs.

Concurrent Process

An instance of running a non-interactive, data-dependent function, simultaneously with online operations. Each time you submit a request, a concurrent manager processes your request, starts a concurrent process, and runs a concurrent program.

Concurrent Program

A concurrent program is an executable file that performs a specific task, such as posting a journal entry or generating a report.

Concurrent Queue

A list of concurrent requests awaiting completion by a concurrent manager. Each concurrent manager has a queue of requests waiting in line to be run. If your system administrator sets up your Oracle Application to have simultaneous queuing, your request can wait to run in more than one queue.

Directory Services

A mapping of Oracle Workflow users and roles to a site's directory repository.

Event

An occurrence in an internet or intranet application or program that might be significant to other objects in a system or to external agents.

Event Activity

A business event modelled as an activity so that it can be included in a workflow process.

Event Data

A set of additional details describing an event. The event data can be structured as an XML document. Together, the event name, event key, and event data fully communicate what occurred in the event.

Event Key

A string that uniquely identifies an instance of an event. Together, the event name, event key, and event data fully communicate what occurred in the event.

Event Message

A standard Workflow structure for communicating business events, defined by the datatype `WF_EVENT_T`. The event message contains the event data as well as several header properties, including the event name, event key, addressing attributes, and error information.

Event Subscription

A registration indicating that a particular event is significant to a system and specifying the processing to perform when the triggering event occurs. Subscription processing can include calling custom code, sending the event message to a workflow process, or sending the event message to an agent.

External Functions

Programs that are executed outside of the Oracle Database.

External Java Functions

Java programs that are executed outside of the Oracle Database by the Java Function Activity Agent.

Function

A PL/SQL stored procedure that can define business rules, perform automated tasks within an application, or retrieve application information. The stored procedure accepts standard arguments and returns a completion result.

Function Activity

An automated unit of work that is defined by a PL/SQL stored procedure.

Generic Service Component Framework

A facility that helps to simplify and automate the management of background Java services.

Item

A specific process, document, or transaction that is managed by a workflow process.

Item Attribute

See Item Type Attribute.

Item Type

A grouping of all items of a particular category that share the same set of item attributes. Item type is also used as a high level grouping for processes.

Item Type Attribute

A feature associated with a particular item type, also known as an item attribute. An item type attribute is defined as a variable whose value can be looked up and set by the application that maintains the item. An item type attribute and its value are available to all activities in a process.

Lookup Code

An internal name of a value defined in a lookup type.

Lookup Type

A predefined list of values. Each value in a lookup type has an internal and a display name.

Message

The information that is sent by a notification activity. A message must be defined before it can be associated with a notification activity. A message contains a subject, a priority, a body, and possibly one or more message attributes.

Message Attribute

A variable that you define for a particular message to either provide information or prompt for a response when the message is sent in a notification. You can use a predefine item type attribute as a message attribute. Defined as a 'Send' source, a message attribute

gets replaced with a runtime value when the message is sent. Defined as a 'Respond' source, a message attribute prompts a user for a response when the message is sent.

Node

An instance of an activity in a process diagram as shown in the Process window.

Notification

An instance of a message delivered to a user.

Notification Activity

A unit of work that requires human intervention. A notification activity sends a message to a user containing the information necessary to complete the work.

Notification Mailer

A type of service component that sends e-mail notifications to users through a mail application, and processes e-mail responses.

Notification Worklist

A Web page that you can access to query and respond to workflow notifications.

Performer

A user or role assigned to perform a human activity (notification). Notification activities that are included in a process must be assigned to a performer.

Process

A set of activities that need to be performed to accomplish a business goal.

Process Definition

A workflow process as defined in Oracle Workflow Builder, which can be saved as a flat file or in a database.

Process Activity

A process modelled as an activity so that it can be referenced by other processes.

Protection Level

A numeric value ranging from 0 to 1000 that represents who the data is protected from for modification. When workflow data is defined, it can either be set to customizable (1000), meaning anyone can modify it, or it can be assigned a protection level that is equal to the access level of the user defining the data. In the latter case, only users operating at an access level equal to or lower than the data's protection level can modify the data.

Result Code

The internal name of a result value, as defined by the result type.

Result Type

The name of the lookup type that contains an activity's possible result values.

Result Value

The value returned by a completed activity.

Role

One or more users grouped by a common responsibility or position.

Service Component Container

An instance of a service or servlet that manages the running of the individual service components that belong to it. The container monitors the status of its components and handles control events for itself and for its components.

Service Component

An instance of a Java program which has been defined according to the Generic Service Component Framework standards so that it can be managed through this framework.

Subscription

See Event Subscription.

System

A logically isolated software environment such as a host machine or database instance.

Timeout

The amount of time during which a notification activity must be performed before the Workflow Engine transitions to an error process or an alternate activity if one is defined.

Transition

The relationship that defines the completion of one activity and the activation of another activity within a process. In a process diagram, the arrow drawn between two activities represents a transition.

Workflow Definitions Loader

A concurrent program that lets you upload and download workflow definitions between a flat file and a database.

Workflow Engine

The Oracle Workflow component that implements a workflow process definition. The Workflow Engine manages the state of all activities for an item, automatically executes functions and sends notifications, maintains a history of completed activities, and detects error conditions and starts error processes. The Workflow Engine is implemented in server PL/SQL and activated when a call to an engine API is made.

Index

Symbols

#WF_REASSIGN_LOV attribute, 3-27
&#NID, 3-11, 3-13, 3-16, 3-45

A

Access level indicator, 3-18
Access property page, 3-17
Access protection
  preserving customizations, 3-18
Acknowledge Ping event, 9-12
ACTID, 6-4, 6-12
Action history, 3-24, 3-26, 3-33
Actions
  for subscriptions, 8-31, 8-61
Activities, 2-7
  accessing from different data stores, 4-4, 5-1
  Concurrent Manager, 5-17
  copy, 3-69
  cost, 3-57
  create, 3-58, 3-60, 3-62, 3-66
  deferred, 3-57
  effective date, 3-68
  error process, 3-68
  event, 3-52, 3-54
  function, 3-52, 3-54
  icons, 3-59, 3-61, 3-63, 3-67, 3-70
  in a loop, 3-69
  in the Buyer: Advanced Shipment Notice process, 10-44
  in the Buyer: Receive Supplier Invoicing process, 10-46
  in the Buyer: Receive Supplier PO Acknowledgement process, 10-42
  in the Buyer: Send PO to Supplier process, 10-40
  in the Buyer: Top Level PO process, 10-37
  in the Default Error Process, 11-6
  in the Default Event Error Process, 11-11
  in the Detail Ping process, 8-103
  in the Master Ping process, 8-102
  in the Notify Approver subprocess, 10-16
  in the Requisition process, 10-11
  in the Retry-only Process, 11-9
  in the Supplier: Advanced Shipment Notice process, 10-58
  in the Supplier: Credit Check process, 10-55
  in the Supplier: Get Order Details process, 10-52
  in the Supplier: Send Supplier Invoice process, 10-60
  in the Supplier: Stock Check process, 10-57
  in the Supplier: Top Level Order process, 10-48
  in the Workflow Event Protocol process, 9-49
  joining branches, 4-3
  notification, 3-52, 3-53
  optional details, 3-68
  process, 3-52, 3-57
  result type, 3-59, 3-61, 3-66
  Standard, 3-52, 5-1
  System: Error, 3-52
  timing out, 4-7
  version number, 3-69
Activity attributes
  Function activity attributes, 3-62, 3-64
  setting values for, 4-9
Activity nodes
  in the Buyer: Advanced Shipment Notice process, 10-44
  in the Buyer: Receive Supplier Invoicing process, 10-46
  in the Buyer: Receive Supplier PO Acknowledgement process, 10-42
  in the Buyer: Send PO to Supplier process, 10-40
  in the Buyer: Top Level PO process, 10-37
  in the Detail Ping process, 8-103
  in the Master Ping process, 8-102
  in the Notify Approver subprocess, 10-16
  in the Requisition process, 10-11
  in the Supplier: Advanced Shipment Notice process, 10-58
  in the Supplier: Credit Check process, 10-55
  in the Supplier: Get Order Details process, 10-52
  in the Supplier: Send Supplier Invoice process, 10-60
  in the Supplier: Stock Check process, 10-57
  in the Supplier: Top Level Order process, 10-48

in the Workflow Event Protocol process, 9-49
Ad hoc users and roles, 4-19
Advanced Queuing, 8-2
Agent Created event, 9-5
Agent Deleted event, 9-6
Agent Group Member Created event, 9-7
Agent Group Member Deleted event, 9-7
Agent Group Member Updated event, 9-7
Agent groups
creating, 8-91
overview, 8-79
updating, 8-91
Agent Updated event, 9-6
Agents, 8-17
creating, 8-89
defining, 8-25
deleting, 8-28
direction, 8-17, 8-77
finding, 8-27
on external systems, 8-87
overview, 8-76
pinging, 8-99
protocol, 8-17, 8-77
queue handlers, 8-19, 8-79
queues, 8-18, 8-78
updating, 8-28, 8-89
viewing, 8-88
Agents Web page, 8-25, 8-28
And activity, 5-2
Any event, 9-14
Any transitions, 4-2
AQ$_JMS_TEXT_MESSAGE, 8-19, 8-79
Arrows, 4-1
Assign activity, 5-12
ATTACHED_URLS attribute, 3-35
Attribute
token substitution, 3-51
Attribute types
attribute, 3-10
date, 3-9, 3-48
document, 3-9, 3-13, 3-49
event, 3-10, 3-49
form, 3-9, 3-12, 3-48
lookup, 3-9, 3-48
number, 3-9, 3-48
role, 3-10, 3-49
text, 3-9, 3-48
URL, 3-9, 3-10, 3-48
Attribute-type attributes, 3-3
Attributes, 3-5
copy, 3-17
type, 3-2, 3-9, 3-48
Automatic replication
of Event Manager objects, 8-48, 8-95

B

B2B Advanced Shipment Notice event, 10-65
B2B Invoice event, 10-67
B2B Purchase Order Acknowledgement event, 10-64
B2B Purchase Order event, 10-61
Block activity, 5-4
Business Event System, 1-2
managing business events, 8-2
Ping/Acknowledge example, 8-99
predefined events, 9-1
Business Event System Applications Control Group, 9-33
Business Event System Control Group, 9-32
Business Event System Control Ping event, 9-32
Business events, 8-3, 8-50
Buyer Workbench
Web page, 10-31
Buyer: Advanced Shipment Notice process
summary, 10-44
Buyer: Receive Supplier Invoicing process
summary, 10-46
Buyer: Receive Supplier PO Acknowledgement process
summary, 10-42
Buyer: Send PO to Supplier process
summary, 10-40
Buyer: Top Level PO process
summary, 10-36

C

Callback functions, 6-10
command, 6-12
for item types, 3-4
Certificate-based digital signatures, 3-30
Compare Date activity, 5-2
Compare Event Property activity, 5-13
Compare Execution Time activity, 5-3
Compare Number activity, 5-2
Compare Text activity, 5-2
Comparison activities, 5-2
Concurrent Manager activities, 5-17
Concurrent Manager Functions item type, 5-17
Content-attached checkbox, 3-50, 3-50
Continue Flow activity, 5-10
Coordinating master/detail activities, 5-9
Cost threshold, 3-57
Customization Level
for activities, 3-7, 3-10, 3-20, 3-45, 3-59, 3-62, 3-64, 3-67, 3-71
Customization level
events, 8-5, 8-52
subscriptions, 8-35, 8-60

D

Date-type attributes, 3-2
Default Error Process, 11-6
Default Event Error Process, 11-11

Default transitions, 4-2
DEFAULT_ERROR, 11-6
DEFAULT_EVENT_ERROR, 11-11
Defer Thread activity, 5-4
Deferred activities, 3-57
Deferred processing
  for event subscriptions, 8-35, 8-66
Demonstration
  directory service, 10-5
Dequeue
  queue handler, 6-22
Detail Ping process
  summary, 8-103
Detail process, 5-9
Developer Studio, 7-2, 7-3
Diagram arrows, 4-1
Digital signatures, 3-30
Directory Service
  in Navigator tree, 2-3
  view from Builder, 4-21
Directory service
  events, 9-38
Dispatch mode, 8-37, 8-69
Document integration, 3-3, 3-9, 3-49, 6-13
Document management integration, 3-3, 3-5
Document message attributes
  attached vs embedded, 3-50
Document-type attributes, 3-3
Documents, 3-5
Dynamic priority, 4-8
Dynamic timeouts, 4-8

E

E-mail notifications
  data type hints, 3-37
  feature, 1-3
Edit menu, B-2
Effective date, 2-12
Effective dates, 2-10, 2-12, 3-68
Effectivity
  dates of, 2-6
Electronic signatures, 3-30
END activities, 4-3
End Activity, 5-7
Enqueue
  queue handler, 6-21
Error handling
  for event subscriptions, 11-3
  for workflow processes, 11-1
Error process, 3-68, 11-5
Event activities, 3-54
  create, 3-62
Event activity attributes, 3-64
Event activity details, 4-9
Event Created event, 9-2
Event data, 6-19, 8-4, 8-30, 8-51, 8-60
Event Deleted event, 9-3
Event for Notification Cancel, 9-22
Event for Notification Close, 9-29
Event for Notification Reassign, 9-22
Event for Notification Respond, 9-29
Event for Notification Send, 9-21
Event Group Creation event, 9-3
Event Group Deleted event, 9-4
Event Group Updated event, 9-4
Event groups, 8-4
  creating, 8-55
  defining, 8-7
  overview, 8-52
  updating, 8-55
Event Manager
  for Oracle Applications, 8-49
  for standalone Oracle Workflow, 8-2
Event nodes, 4-9
Event subscriptions, 8-29, 8-58
  rule functions, 6-23
Event Subscriptions Web page, 8-43
Event System Demonstration
  data model, 10-30
  initiating, 10-31
  overview, 10-28
  setting up, 10-31
Event System Demonstration process
  installing, 10-30
Event Updated event, 9-2
Event-type attributes, 3-3
Events, 8-3
  creating, 8-54
  defining, 8-5
  deleting, 8-11
  finding, 8-10
  overview, 8-50
  predefined, 9-1
  raising, 8-4, 8-44, 8-49
  sending to agents, 8-34, 8-63
  sending to workflow processes, 8-32, 8-61
  testing, 8-57
  updating, 8-11, 8-54
  viewing, 8-53
Events Web page, 8-5, 8-7, 8-11, 8-38
Events: Buyer Workbench
  Web page, 10-31
Events: Track Order
  Web page, 10-33
Example function activity
  Select Approver, 10-21
  Verify Authority, 10-24
Example process
  Event System Demonstration, 10-28
  Requisition, 10-3
Execute Concurrent Program activity, 5-17
Expand Roles check box, 3-53, 3-59, 3-70
External document integration, 3-5
External system registration, 8-98

F

File menu, B-1
Find Agent Web page, 8-27
Find Event Web page, 8-10, 8-42
Find System Web page, 8-15
Fonts
  modifying, 4-18
Form-type attributes, 3-2
Frame target
  URL attributes, 3-49
Framework, 3-38
FROM_ROLE attribute, 3-29
FUNCMODE, 6-4
Function activities, 3-54
  create, 3-60
  standard Java API, 6-7
  standard PL/SQL API, 6-2
Function activity attributes, 3-7, 3-62
Functions, 2-8
  PL/SQL procedures, 2-8
Future dated events, 8-64
Future-dated events, 8-36, 8-67

G

Generate function, 8-4, 8-51
Generic Service Component Framework
  events, 9-34
Get Event Property activity, 5-12
Get Monitor URL activity, 5-12
Global variables, 3-2
GSC Business Event System Control Group, 9-36

H

HDR attributes, 3-31
HDR_REGION attribute, 3-32
Header attributes, 3-31
Header region, 3-32
Help menu, B-4
Hidden item types, 2-3
HIDE_MOREINFO attribute, 3-28
HIDE_REASSIGN attribute, 3-27
HISTORY attribute, 3-33

I

Icons
  viewing, 3-59, 3-61, 3-63, 3-67
Images
  in notifications, 3-11
Initiating a workflow process, 10-6, 10-31
Item attributes
  external document integration, 3-5
Item type attributes, 3-2, 3-7, 3-7
  Event System Demonstration, 10-34
  Requisition, 10-8
  Workflow Send Protocol, 9-46
Item types, 2-7, 3-2
  callback function, 3-4
  Concurrent Manager Functions, 5-17
  context reset, 6-10
  copy, 3-16
  creation, 3-6
  Event System Demonstration, 10-34
  loading, 2-9, 2-9
  persistence type, 3-4
  Requisition, 10-8
  saving, 2-9
  selector functions, 3-4, 6-10
  Standard, 5-1
  System: Error, 11-5
  Workflow Agent Ping/Acknowledge, 8-100
  Workflow Send Protocol, 9-46
ITEMKEY, 6-4, 6-12
ITEMTYPE, 6-4, 6-11

J

Java API
  for function activities, 6-7
Java APIs
  for a queue handler, 6-21
JDeveloper, 3-38
JMS Text messages, 8-19, 8-79
Joining activities, 4-3

L

Launch Process activity, 5-5
Launch Summary Notifications event, 9-30
Licensing, 8-52, 8-60
List of values
  in a Web interface, 8-17
Load balancing, 5-7
Loading item types, 2-9
Local system, 8-12, 8-92
Local System Identifier, 8-97
Lookup codes
  copy, 3-22
Lookup types, 2-7, 3-19, 3-20
  copy, 3-22
  creation, 3-20
Lookup-type attributes, 3-2
Loop Counter activity, 5-5
Loop Reset, 4-2
Loops, 3-69, 6-5

M

Master Ping Process
  summary, 8-101
Master process, 5-9
Master/copy systems, 8-49, 8-95
Master/Detail coordination activities, 5-9
  notes on usage, 5-11

Menus
  Oracle Workflow Builder, B-1
Message attributes, 3-23, 3-23, 3-46, 3-46, 10-27
  #ATTACHED_URLS, 3-35
  #FROM_ROLE, 3-29
  #HDR, 3-31
  #HDR_REGION, 3-32
  #HIDE_MOREINFO, 3-28
  #HIDE_REASSIGN, 3-27
  #HISTORY, 3-33
  #PERZ_FUNCTION_NAME, 3-33
  #PERZ_LOCALIZATION_CODE, 3-34
  #PERZ_ORGANIZATION_ID, 3-33
  #RELATED_APPL, 3-32
  #SUBMIT_COMMENTS, 3-33
  #WF_REASSIGN_LOV, 3-27
  #WF_SECURITY_POLICY, 3-31
  #WF_SIG_ID, 3-31
  #WF_SIG_POLICY, 3-30
  #WFM_CANCELED, 3-35
  #WFM_CLOSED, 3-36
  #WFM_FROM, 3-34
  #WFM_HTMLAGENT, 3-34
  #WFM_LANGUAGE, 3-34
  #WFM_OPEN_INVALID, 3-36
  #WFM_OPEN_MAIL, 3-35
  #WFM_OPEN_MAIL_DIRECT, 3-35
  #WFM_OPEN_MAIL_FYI, 3-35
  #WFM_OPEN_MORE_INFO, 3-36
  #WFM_REPLYTO, 3-34
  #WFM_TERRITORY, 3-34
  formatted table, 3-26
  message template attributes, 3-35
  notification mailer attributes, 3-34
  personalization attributes, 3-33
  Respond, 3-24, 3-36, 3-47
  Send, 3-23, 3-47
  source, 3-23, 3-47
  WF_NOTE, 3-25
Message function
  WF_NOTIFICATION(), 3-26
Message templates, 3-35
Messages, 2-7, 3-22
  body, 3-43, 10-26
  copy, 3-52
  creation, 3-42
  overriding default priority, 4-8
  subject, 3-43, 10-26
  viewing, 10-27
Monitoring
  work items, 1-3
Multi-consumer queues, 8-34, 8-63

N

Naming conventions
  PL/SQL stored procedures, 10-11
Navigation paths, A-1
Navigator Toolbar, B-4
Navigator tree
  finding objects in, 2-4
Node activities
  dynamic priority, 4-8
Nodes
  adding to a process, 4-4
  start and end, 4-5
NOOP activity, 5-5
Notification activities, 3-53
  coupling with custom functions, 3-59
  create, 3-58
  Notify Requisition Approval Required, 10-26
Notification Cancel event, 9-22
Notification Close event, 9-29
Notification Error event, 9-27
Notification functions, 3-59
Notification history, 3-26
Notification ID token, 3-11, 3-13, 3-16, 3-45
Notification mailers
  events, 9-30
Notification Reassign event, 9-22
Notification Receive Group, 9-27
Notification Recipient Is Unavailable event, 9-27
Notification Respond event, 9-29
Notification Send Error event, 9-25
Notification Send event, 9-21
Notification Send Group, 9-23
Notification Web page, 1-3
Notifications
  action history, 3-33
  e-mail attributes, 3-34
  header attributes, 3-31
  header region, 3-32
  hiding the Reassign button, 3-27
  hiding the Request Information button, 3-28
  load balancing, 5-7
  region personalizations, 3-33
  Related Applications region, 3-32
  requiring an electronic signature, 3-30
  security policy, 3-31
  setting the From Role, 3-29
  specifying a list of users for reassignment, 3-27
  specifying message templates, 3-35
Notify activity, 5-8
Notify Approver
  example notification activities, 10-26
Notify Approver subprocess
  summary, 10-15
Notify Requisition Approval Required, 10-26
Number-type attributes, 3-2

O

Or activity, 5-2
Oracle Advanced Queuing, 8-2
Oracle Applications Framework
  embedding regions in notifications, 3-38

Oracle Applications Manager, 1-4
Oracle JDeveloper, 3-38
Oracle Workflow Builder, 1-2
  Loader functionality, 2-11
  overview, 2-1
  save modes, 2-11, 3-18
  starting from command line, 2-13
Oracle Workflow Manager, 1-4
oracle.apps.wf.engine.abort event, 9-37
oracle.apps.wf.engine.skip event, 9-38

P

Password-based signatures, 3-30
Persistence, 3-4
PERZ_FUNCTION_NAME attribute, 3-33
PERZ_LOCALIZATION_CODE attribute, 3-34
PERZ_ORGANIZATION_ID attribute, 3-33
Phase numbers, 8-30, 8-37, 8-59, 8-68
Ping Agent event, 9-12
Pinging agents, 8-99
PL/SQL, 1-3
  document, 6-13, 6-13
PL/SQL APIs
  for a queue handler, 6-21
  for a selector or callback function, 6-10
  for a 'PL/SQL BLOB' document, 6-13
  for a 'PL/SQL CLOB' document, 6-13
  for a 'PL/SQL' document, 6-13
  for an Event Data Generate Function, 6-19
  for an Event Subscription Rule Function, 6-23
  for function activities, 6-2
PL/SQL BLOB
  document, 6-13, 6-18
PL/SQL CLOB
  document, 6-13, 6-15
PL/SQL documents, 3-5
PL/SQL stored procedures
  creating, 10-11
  naming conventions, 10-11
  scripts, 10-11
Post-notification functions, 3-53
Predefined events, 9-1
Preserving customizations
  for an activity, 3-18
Process activities, 3-57
  create, 3-66
Process definition
  modifying, 2-8
Process diagram
  adding nodes, 4-4
  drawing, 4-1, 4-4
Process window
  editing, 4-1
Process Window Toolbar, B-6
Processes
  activity transitions, 4-1
  copying to clipboard, 4-17
  creation, 2-5
  editing, 2-7, 2-9
  loops, 6-5
  overview, 4-15
  printing, 4-16
  starting, 4-22
  verify, 4-17
Protocols, 8-17, 8-77

Q

Queue handlers, 6-21, 8-19, 8-79
Queues
  assigned to agents, 8-18, 8-78

R

Raise Event Web page, 8-44
Raising events, 8-4, 8-44, 8-49
Reassign notifications
  hiding the Reassign button, 3-27
  specifying a list of users for reassignment, 3-27
Receipt of Incoming Response event, 9-26
Refresh event, 9-35
Related Applications region, 3-32
RELATED_APPL attribute, 3-32
Requisition
  data model, 10-4
Requisition Demonstration
  Web page, 10-6
Requisition process, 10-3
  example function activities, 10-21
  initiating, 10-6
  installing, 10-4
  summary, 10-9
RESULT, 6-4, 6-12
Result type
  for activities, 3-59, 3-61, 3-66
  for voting activities, 3-70
Resume event, 9-35
Retry Error, 11-9
RETRY_ONLY, 11-9
Return to Sender event, 9-26
Role
  property page, 4-21
Role Resolution activity, 5-7
Role-type attributes, 3-3
Roles, 4-19
  ad hoc, 4-19
  loading into the Workflow Builder, 4-19
  tab page, 4-19
  view from Builder, 4-21
Rule functions, 6-23
  for event subscriptions, 8-31, 8-65
Run Workflow page, 7-3
Running test processes, 7-3

S

Sample workflow processes, 10-1
Savepoints, 6-2, 6-7
Security policy
  for e-mail notifications, 3-31
Seed event group, 9-9
Select Approver function activity, 10-21
Selector functions, 3-4, 6-10
Send Summary Notification event, 9-25
Set Event Property activity, 5-13
SetItemParent API, 5-9
Shortcuts, 4-18
Signature IDs, 3-31
Signatures, 3-30
Single-consumer queues, 8-35, 8-63
Source types, 8-29, 8-59
Stack Layout regions, 3-40
Standard activities, 5-1
Standard APIs
  for "PL/SQL BLOB" documents, 6-13, 6-18
  for "PL/SQL CLOB" documents, 6-13, 6-15
  for "PL/SQL" documents, 6-13, 6-13
  for a Queue Handler, 6-21
  for an Event Data Generate Function, 6-19
  for an Event Subscription Rule Function, 6-23
  for function activities, 6-2, 6-7
  for selector/callback functions, 6-10
Standard error process, 11-5
Standard item type, 5-1
START activities, 4-3
Start activity, 5-7
Start event, 9-34, 9-38
StartProcess function
  for sample Requisition process, 10-18
Stop event, 9-36
Submit Concurrent Program activity, 5-18
SUBMIT_COMMENTS attribute, 3-33
Subprocesses
  timing out, 4-7
Subscription Created event, 9-8
Subscription Deleted event, 9-8
Subscription processing, 8-61
Subscription Updated event, 9-8
Subscriptions, 8-29
  creating, 8-71
  deferring, 8-35, 8-66
  defining, 8-38
  deleting, 8-43
  finding, 8-42
  overview, 8-58
  predefined, 9-1
  updating, 8-43, 8-71
  viewing, 8-70
Supplier: Advanced Shipment Notice process
  summary, 10-58
Supplier: Credit Check process
  summary, 10-54
Supplier: Get Order Details process
  summary, 10-51
Supplier: Send Supplier Invoice process
  summary, 10-59
Supplier: Stock Check process
  summary, 10-56
Supplier: Top Level Order process
  summary, 10-47
Suspend event, 9-35
Synchronize Event Systems event, 9-9
System Created event, 9-4
System Deleted event, 9-5
System identifier, 8-46
System Identifier Web page, 8-46
System integration, 8-2
System Signup event, 9-13
System Signup Web page, 8-47
System Updated event, 9-5
System: Error item type, 11-5
Systems, 8-12
  creating, 8-97
  defining, 8-13
  deleting, 8-16
  external, 8-93
  finding, 8-15
  local, 8-12, 8-92
  local system identifier, 8-97
  master/copy, 8-49, 8-95
  overview, 8-92
  registering Workflow-enabled systems, 8-98
  signing up, 8-45, 8-47, 8-93
  synchronizing, 8-47, 8-94
  updating, 8-16, 8-97
  viewing, 8-96
Systems Web page, 8-13, 8-16

T

Testing workflow definitions, 7-3
Text-type attributes, 3-2
Timeout transitions, 4-2, 4-2
Timeouts, 4-7
  dynamic, 4-8
Token substitution
  attributes, 3-51
  of document-type message attributes, 3-13
Toolbars
  Oracle Workflow Builder, B-4
Transitions, 4-1
  Any, 4-2
  creating, 4-14
  Default, 4-2
  editing, 4-14
  Timeout, 4-2

U

Unexpected event, 9-17, 11-5

Unsolicited Email Threshold Reached event, 9-31
URL attributes
  frame target, 3-49
URL message attributes
  attached vs embedded, 3-50
URL-type attributes, 3-3
URLs
  for Event System Demonstration Web pages, 10-31, 10-33
  for Requisition Demonstration Web page, 10-7
User Entry Has Changed event, 9-19
Users
  ad hoc, 4-19

V

Verify Authority function activity, 10-24
Version number
  for activities, 3-69
Versioning, 2-6
View menu, B-3
Vote Yes/No activity, 5-8
Voting activities
  result type, 3-70
Voting activity, 3-69

W

Wait activity, 5-3
Wait for Concurrent Program activity, 5-19
Wait for Flow activity, 5-10
WF_CONTROL agent, 8-19, 8-79
WF_DEFERRED agent, 8-19, 8-79
WF_ERROR agent, 8-19, 8-80
WF_ERROR_QH, 8-19, 8-79
WF_EVENT_OJMSTEXT_QH, 8-19, 8-79
WF_EVENT_QH, 8-19, 8-79
WF_IN agent, 8-19, 8-80
WF_JAVA_DEFERRED agent, 8-80
WF_JAVA_ERROR agent, 8-80
WF_JMS_IN agent, 8-19, 8-80
WF_JMS_OUT agent, 8-19, 8-80
WF_NOTE attribute, 3-25
WF_NOTIFICATION() message function, 3-26
WF_NOTIFICATION_IN agent, 8-19, 8-80
WF_NOTIFICATION_OUT agent, 8-19, 8-80
WF_OUT agent, 8-19, 8-80
WF_REASSIGN_LOV attribute, 3-27
WF_REQDEMO.SelectApprover, 10-21
WF_REQDEMO.StartProcess, 10-6
WF_REQDEMO.VerifyAuthority, 10-14, 10-24
WF_SECURITY_POLICY attribute, 3-31
WF_SIG_ID attribute, 3-31
WF_SIG_POLICY attribute, 3-30
WF_WS_JMS_IN agent, 8-80
WF_WS_JMS_OUT agent, 8-80
WFM_CANCELED attribute, 3-35
WFM_CLOSED attribute, 3-36
WFM_FROM attribute, 3-34
WFM_HTMLAGENT attribute, 3-34
WFM_LANGUAGE attribute, 3-34
WFM_OPEN_INVALID attribute, 3-36
WFM_OPEN_MAIL attribute, 3-35
WFM_OPEN_MAIL_DIRECT attribute, 3-35
WFM_OPEN_MAIL_FYI attribute, 3-35
WFM_OPEN_MORE_INFO attribute, 3-36
WFM_REPLYTO attribute, 3-34
WFM_TERRITORY attribute, 3-34
WFRUND.SQL, 10-6
Window menu, B-4
Work items, 2-7
Workflow Agent Ping/Acknowledge, 8-99
  item type, 8-100
  item type attributes, 8-100
Workflow Builder, 1-2
Workflow Builder menus, B-1
Workflow definitions
  loading, 1-3
  source control, 2-9
  testing, 7-1, 7-2
Workflow Definitions Loader, 1-3
Workflow Demonstrations home page, 10-1
Workflow diagrams
  displaying, 10-2
Workflow Engine, 1-2
  cost threshold, 3-57
  error processing, 11-1
  events, 9-37
Workflow Event Protocol process
  summary, 9-47
Workflow processes
  samples, 10-1
Workflow Role Hierarchy Relationship Created event, 9-43
Workflow Role Hierarchy Relationship Updated event, 9-44
Workflow Role Updated event, 9-39
Workflow Send Protocol
  item type, 9-46
  sample workflow process, 9-45
Workflow Send Protocol Acknowledgement event, 9-53
Workflow Send Protocol event, 9-51
Workflow User Updated event, 9-38
Workflow User/Role Relationship Created event, 9-40
Workflow User/Role Relationship Updated event, 9-42

X

X.509 certificates, 3-30
XML Compare Tag Value (Date) activity, 5-15
XML Compare Tag Value (Number) activity, 5-15
XML Compare Tag Value (Text) activity, 5-15
XML Compare Tag Value activities, 5-15

XML Get Tag Value activity, 5-14
XML Transform activity, 5-16