



Oracle Workflow™ Guide

Release 2.0.3

Part No. A56104-01

ORACLE®

Enabling the Information Age™

Oracle Workflow Guide, Release 2.0.3

Part No. A56104-01

Copyright © 1997 Oracle Corporation

All rights reserved.

Primary Author: Siu Chang

Contributors: George Buzsaki, George Kellner, Rama Kocherlakota, Kevin Hudson, David Lam, Jin Liu, Tim Roveda, Robin Seiden, Sheryl Sheh, Susan Stratton

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

This Program contains proprietary information of Oracle Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright patent and other intellectual property law. Reverse engineering of the software 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. Oracle Corporation does not warrant that this document is error free. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If this Program is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

Restricted Rights Legend Programs delivered subject to the DOD FAR Supplement are 'commercial computer software' and use, duplication and disclosure of the Programs shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are 'restricted computer software' and use, duplication and disclosure of the Programs shall be subject to the restrictions in FAR 52..227-14, Rights in Data -- General, including Alternate III (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

Oracle, Oracle Alert, Oracle Application Object Library, Oracle Financials, SQL*Net, and SQL*Plus are registered trademarks of Oracle Corporation.

Oracle7, Oracle8, Oracle Applications, Oracle Data Browser, Oracle Human Resources, Oracle InterOffice, Oracle Installer, Oracle Office, Oracle Payables, Oracle Projects, Oracle Purchasing, Oracle Service, Oracle Web Application Server, Oracle Web Customers, Oracle Web Employees, Oracle WebServer, Oracle Web Suppliers, Oracle Workflow, and PL/SQL are trademarks of Oracle Corporation.

All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.



Contents

- Preface** v
 - About This User's Guide vi
 - Assumptions vii
 - Do Not Use Database Tools to Modify Oracle Applications Data viii
 - Other Information Sources viii
 - Related User's Guides ix
 - About Oracle xii
 - Thank You xii

- Chapter 1** **Overview of Oracle Workflow** 1 – 1
 - Introduction to Oracle Workflow 1 – 2
 - Major Features and Definitions 1 – 3
 - Workflow Processes 1 – 6

- Chapter 2** **Setting Up Oracle Workflow** 2 – 1
 - Oracle Workflow Hardware and Software Requirements 2 – 2
 - Implementing Oracle Workflow 2 – 4
 - Required Set Up Steps 2 – 4
 - Optional Set Up Steps 2 – 5
 - Other Workflow Features 2 – 5
 - Identifying the Version of Your Oracle Workflow Server ... 2 – 6
 - Setting Up an Oracle Workflow Directory Service 2 – 7
 - Predefined Directory Services 2 – 10

Creating the WF_LANGUAGES View	2 – 13
Setting the WF_RESOURCES Environment Variable	2 – 14
Identifying the Oracle Web Agent used by Oracle Workflow ...	2 – 15
Identifying the Oracle Workflow Administration Role	2 – 18
Setting Up the Workflow Monitor and Oracle Workflow's Web Pages	2 – 19
Secure the Workflow Database Connection Descriptor (DCD) ..	2 – 21
Implementing the Notification Mailer	2 – 25
Modifying Your Message Templates	2 – 36
Setting Up Background Workflow Engines	2 – 43
Adding Custom Icons to Oracle Workflow	2 – 47
Overview of Oracle Workflow Access Protection	2 – 48
Setting Up a Default Access Level	2 – 52
Using the Workflow Definitions Loader	2 – 54

Chapter 3

Defining Workflow Process Components	3 – 1
Overview of Oracle Workflow Builder	3 – 2
The Navigator Tree Structure	3 – 3
Viewing the Navigator Tree	3 – 4
Creating Process Definitions in Oracle Workflow Builder	3 – 6
Opening and Saving Item Types	3 – 8
Item Types	3 – 14
Allowing Access to an Object	3 – 24
Lookup Types	3 – 26
Messages	3 – 30
Activities	3 – 41

Chapter 4

Defining a Workflow Process Diagram	4 – 1
Process Window	4 – 2
Modifying Fonts in Oracle Workflow Builder	4 – 13
Creating a Shortcut Icon for a Workflow Process	4 – 14
Roles	4 – 16

Chapter 5

Predefined Workflow Activities	5 – 1
Standard Activities	5 – 2
Default Error Process	5 – 16

Chapter 6	Defining PL/SQL Procedures for Oracle Workflow 6 – 1
	Standard API for PL/SQL Procedures Called by Function Activities 6 – 2
	Standard API for an Item Type Selector or Callback Function . . 6 – 5
	Standard API for a "PL/SQL" Document 6 – 9
Chapter 7	Oracle Workflow APIs 7 – 1
	Oracle Workflow Procedures and Functions 7 – 2
	Overview of the Workflow Engine 7 – 3
	Workflow Engine APIs 7 – 8
	Workflow Core APIs 7 – 34
	Workflow Purge APIs 7 – 42
	Workflow Directory Services APIs 7 – 49
	Workflow Monitor APIs 7 – 58
	Oracle Workflow Views 7 – 62
	Overview of Notification APIs 7 – 67
	Notification Model 7 – 67
	Notification APIs 7 – 69
Chapter 8	Viewing Notifications and Processing Responses 8 – 1
	Overview of Notification Handling 8 – 2
	Reviewing Notifications in the Notification Viewer (for Oracle Applications Users Only) 8 – 2
	Reviewing Notifications via Electronic Mail 8 – 6
	Viewing Notifications from a Web Browser 8 – 12
	Reviewing a Summary of Your Notifications via Electronic Mail 8 – 18
	Defining Rules for Automatic Notification Handling 8 – 20
	Accessing the Oracle Workflow Home Page 8 – 27
Chapter 9	Monitoring Workflow Processes 9 – 1
	Overview of Workflow Monitoring 9 – 2
	Workflow Status Form 9 – 2
	Workflow Monitor 9 – 3
	Workflow Monitor Access 9 – 8
Chapter 10	Sample Workflow: Requisition Approval Process 10 – 1
	Requisition Approval Process 10 – 2

Installing the Demonstration Data Model	10 – 3
Displaying a Process Window	10 – 5
The Workflow Demonstration Item Type	10 – 6
Summary of the Requisition Approval Process	10 – 8
Requisition Approval Process Activities	10 – 10
Summary of the Notify Approver Subprocess	10 – 15
Notify Approver Subprocess Activities	10 – 16
Initiating the Requisition Approval Workflow	10 – 18
Example Function Activities	10 – 26
Example: Select Approver	10 – 26
Example: Verify Authority	10 – 29
Example Notification Activity	10 – 32
Example: Notify Requisition Approval Required	10 – 32

Chapter 11	Workflow Administration Scripts	11 – 1
	Miscellaneous SQL Scripts	11 – 2

Appendix A	Oracle Workflow Builder Menus and Toolbars	A – 1
	Oracle Workflow Builder Menus	A – 2
	Oracle Workflow Builder Toolbars	A – 6

Appendix B	Oracle Applications Embedded Workflows	B – 1
	Predefined Workflows Embedded in Oracle Applications and Oracle Self-Service Web Applications	B – 2

Glossary

Index



Preface

Welcome to the *Oracle Workflow Guide*.

This guide includes the information you need to work with Oracle Workflow effectively. It contains detailed information about the following:

- Overview and reference information
- Oracle Workflow implementation suggestions
- Oracle Workflow functions and features

This preface explains how this guide is organized and introduces other sources of information that can help you.

About This User's Guide

This guide is the primary source of information about Oracle Workflow. It contains overviews as well as task and reference information. This guide includes the following chapters:

- Chapter 1 provides an overview of Oracle Workflow.
- Chapter 2 describes how to implement Oracle Workflow for your site.
- Chapter 3 describes how to define the components necessary to build a workflow process.
- Chapter 4 describes how to draw and define a workflow process diagram.
- Chapter 5 describes the standard activities provided with Oracle Workflow.
- Chapter 6 describes the standard APIs for the PL/SQL functions that can be called by Oracle Workflow.
- Chapter 7 provides detailed information about Oracle Workflow's APIs.
- Chapter 8 discusses how a user can view and act on a workflow notification.
- Chapter 9 describes how to use the Workflow Monitor to administer or view the status of a workflow process.
- Chapter 10 describes the demonstration workflow process included with the standalone version of Oracle Workflow.
- Chapter 11 describes the miscellaneous administrative SQL scripts included with Oracle Workflow.
- Appendix A describes the Oracle Workflow Builder menus and toolbar.
- Appendix B lists the predefined workflow processes that are included with the Oracle Applications–embedded version of Oracle Workflow.

This guide is available online

The paper and online versions of this manual have identical content; use whichever format is most convenient.

If you are using the version of Oracle Workflow embedded in Oracle Applications, note that this guide is available online, in Windows Help, HTML and Adobe Acrobat format.

The Windows Help version of this book is optimized for onscreen reading, and lets you follow hypertext links to various topics. The Windows Help is available from the Oracle Workflow Builder Help menu.

The HTML version of this book is optimized for onscreen reading, and lets you follow hypertext links for easy access to books across our entire library. The HTML documentation is available from the Oracle Applications toolbar, or from a URL provided by your system administrator.

You can also order an Oracle Applications Documentation Library CD containing Adobe Acrobat versions of each manual in the Oracle Applications documentation set. Using this CD, you can search for information, read it onscreen, and print individual pages, sections, or entire books. When you print from Adobe Acrobat, the resulting printouts look just like pages from an Oracle Applications hardcopy manual.

Note: There may be additional material that was not available when this user's guide was printed. To learn if there is a documentation update for this product, look at the main menu on this product's HTML help.

If you are using the standalone version of Oracle Workflow, note that this guide is available online, in both Windows Help and HTML format.

The Windows Help is available from the Oracle Workflow Builder Help menu.

The HTML documentation is available from a URL provided by your system administrator.

Assumptions

This guide assumes you have a working knowledge of the principles and customary practices of your business area. If you have never used Oracle Workflow, we suggest you attend an Oracle Workflow training class available through Oracle Education. (See Other Information Sources for more information about Oracle Workflow and Oracle training.)

This guide also assumes you have a basic understanding of operating system concepts and familiarity with Oracle7 or Oracle8, PL/SQL, and Oracle WebServer/Oracle Application Server technology. If you have not yet been introduced to any of these systems, we suggest you attend one or more classes available through Oracle Education.

Do Not Use Database Tools to Modify Oracle Applications Data

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 like SQL*Plus to modify Oracle Workflow data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle Workflow tables are interrelated, any change you make using an Oracle Workflow user interface or API can update many tables at once. But when you modify Oracle Workflow data using anything other than an Oracle Workflow user interface or API, 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 Workflow.

When you use Oracle Workflow user interfaces or APIs to modify your data, Oracle Workflow automatically checks that your changes are valid. But, if you enter information into database tables using database tools, you may store invalid information.

Consequently, we STRONGLY RECOMMEND that you never use SQL*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle Workflow tables, unless we tell you to do so in our manuals.

Other Information Sources

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

If you are using the version of Oracle Workflow embedded in Oracle Applications, most Oracle Applications documentation is available in Adobe Acrobat format on the *Oracle Applications Documentation Library* CD. We supply this CD with every software shipment.

If this manual refers you to other Oracle Applications documentation, use only the Release 11 versions of those manuals unless we specify otherwise.

Related User's Guides

Oracle Workflow is used by other Oracle Applications products to provide embedded workflows. Therefore, you may want to refer to other user's guides to learn more about the embedded workflows.

If you do not have the hardcopy versions of these manuals, you can read them by choosing Library from the Help menu, or by reading from the Oracle Applications Document Library CD, or by using a web browser with a URL that your system administrator provides.

Oracle General Ledger User's 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. It 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 Purchasing User's Guide

If you install Oracle Purchasing, refer to this user's guide to read about entering and managing the purchase orders to which you match invoices.

Oracle HRMS User's Guide

This manual explains how to enter your employees, so you can enter expense reports for them. It also explains how to set up organizations and site locations.

Oracle Receivables User's Guide

Use this manual to learn how to implement flexible address formats for different countries. You can use flexible address formats in the suppliers, banks, invoices, and payments windows.

Oracle Projects User's Guide

If you install Oracle Projects, use this user's guide to learn how to enter expense reports in Projects that you import into Payables to create invoices. You can also use this manual to see how to create Project information in Projects which you can then record for an invoice or invoice distribution.

Oracle Financials Open Interfaces Guide

This guide is a compilation of all open interface discussions in all Oracle Financial Applications user's guides.

Oracle Applications Implementation Wizard User's Guide

If you are implementing more than one Oracle product, you can use the Oracle Applications Implementation Wizard to coordinate your setup activities. This guide describes how to use the wizard.

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*. It also provides information to help you build your custom Developer/2000 forms so that they integrate with Oracle Applications.

Installation and System Administration

Oracle Applications Installation Manual

This manual and the accompanying release notes provide information you need to successfully install Oracle Financials, Oracle Public Sector Financials, Oracle Manufacturing, or Oracle Human Resources in your specific hardware and operating system software environment.

Oracle Applications Upgrade Manual

This manual explains how to prepare your Oracle Applications products for an upgrade. It also contains information on finishing the upgrade procedure for each product. Refer to this manual and the *Oracle Applications Installation Manual* when you plan to upgrade your products.

Oracle Applications System Administrator's Guide

This manual 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 processing.

Oracle Application Object Library Technical Reference Manual

The *Oracle Applications Object Library Technical Reference Manual* contains database diagrams and a detailed description of Oracle Application Object Library and Oracle Workflow and related applications database tables, forms, reports, and programs.

You can order a technical reference manual for any product you have licensed. Technical reference manuals are available in paper format only.

Other Information

Training

Oracle Education offers a complete set of training courses to help you and your staff master Oracle Applications. We can help you develop a training plan that provides thorough training for both your project team and your end users. We will work with you to organize courses appropriate to your job or area of responsibility.

Training professionals can show you how to plan your training throughout the implementation process so that the right amount of information is delivered to key people when they need it the most. You can attend courses at any one of our many Educational Centers, or you can arrange for our trainers to teach at your facility. In addition, we can tailor standard courses or develop custom courses to meet your needs.

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 server, and your hardware and software environment.

About Oracle

Oracle Corporation develops and markets an integrated line of software products for database management, applications development, decision support, and office automation, as well as Oracle Applications, an integrated suite of more than 45 software modules for financial management, supply chain management, manufacturing, project systems, human resources, and sales and service management.

Oracle products are available for mainframes, minicomputers, personal computers, network computers, and personal digital assistants, allowing organizations to integrate different computers, different operating systems, different networks, and even different database management systems, into a single, unified computing and information resource.

Oracle is the world's leading supplier of software for information management, and the world's second largest software company. Oracle offers its database, tools, and applications products, along with related consulting, education, and support services, in over 140 countries around the world.

Thank You

Thank you for using Oracle Workflow and this guide.

We value your comments and feedback. At the end of this manual is a Reader's Comment Form you can use to explain what you like or dislike about Oracle Workflow or this user's guide. Mail your comments to the following address or call us directly at (650) 506-7000.

Oracle Applications Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Or, send electronic mail to appsdoc@us.oracle.com.

Overview of Oracle Workflow

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

- Oracle Workflow Builder, a graphical tool that lets you create business process definitions.
- The Workflow Engine, which implements process definitions at runtime.
- The Notifications System, which sends notifications to and processes responses from users in a workflow.
- Graphical Monitoring Tool, which allows you to track your workflow process using a Web Browser.

Introduction to Oracle Workflow

Business processes today involve getting many types of information to multiple people according to rules that are constantly changing. Oracle Workflow lets you automate and continuously improve business processes, routing information of any type according to business rules you can easily change to people both inside and outside your enterprise. See: Major Features and Definitions: page 1 – 3.

Routing Information

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 act. Oracle Workflow can route supporting information to each decision maker in a business process.

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 full power of PL/SQL, the language of the Oracle7 Server, to express any business rule that affects a workflow process. See: Workflow Processes: page 1 – 6.

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 respond using a standard Web browser or an Oracle Applications Notification form.

Major Features and Definitions

Oracle Workflow Builder

Oracle Workflow Builder 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 – 6.

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 Oracle7 server 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. 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.

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 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. 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 associated with it, which contains all the information a user needs to make a decision, as well as possible responses. Oracle Workflow interprets each response and moves on to the next workflow activity.

Personal Inbox

Users who connect to Oracle Applications can see all the notifications awaiting their attention in a common Notification Viewer form, or Personal Inbox. Choosing a notification takes users to a Notification Details window that describes any actions they need to take. The Notification Details window can take users directly to an Oracle Applications form where they can perform the necessary action.

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 HTML 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

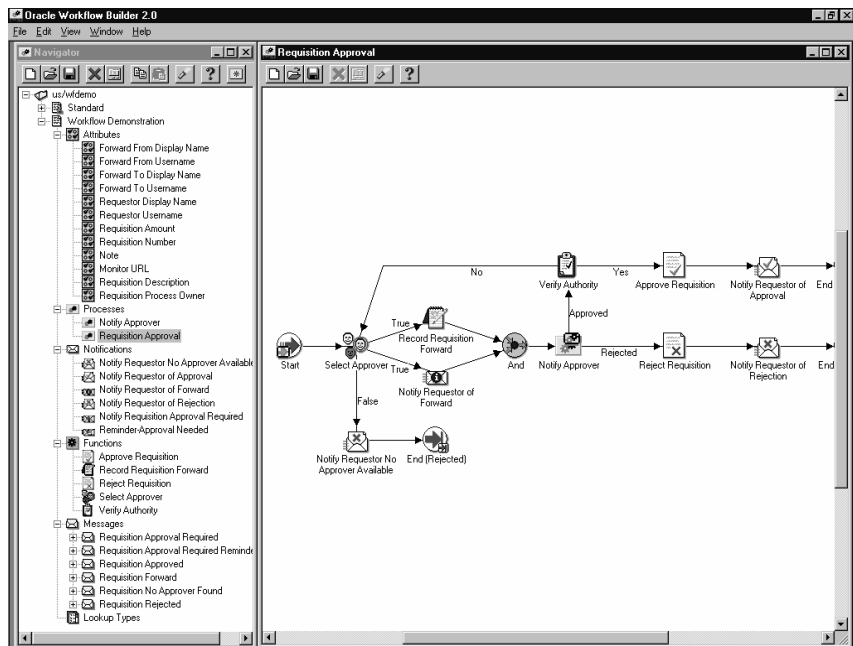
also displays a separate status summary for the work item, the process, and each activity in the process.

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, a notification to a user or role that may optionally request a response, 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.



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.

CHAPTER

2

Setting Up Oracle Workflow

This chapter describes the requirements for Oracle Workflow and provides the steps necessary to set up Oracle Workflow at your site.

Oracle Workflow Hardware and Software Requirements

The components of Oracle Workflow require the following hardware and software configurations:

- Oracle Workflow Builder is installed using Oracle Installer and requires the installation of SQL*Net V2 (included). You should install Oracle Workflow Builder on an IBM, Compaq or 100% compatible personal computer with the following:
 - A 486 processor or better
 - Clock speed of 66 Mhz or greater (90 Mhz or greater is recommended)
 - Network card
 - SVGA color monitor
 - Modem configured with dial-in access for use by Oracle Worldwide Customer Support. At least one PC at your site should be configured with a modem.
 - Dual speed, ISO 9660 format CD-ROM available as a logical drive
 - Microsoft Windows 95 or Windows NT
 - At least 17.3 MB of available disk space to install Oracle Installer, Oracle Workflow Builder, and SQL*Net V2.



Attention: SQL*Net requires and only supports the use of Microsoft's TCP/IP drivers.

- The Workflow Server requires Oracle WebServer to be previously installed.
- The Notification Viewer form, also known as the "Personal Inbox", runs on any Oracle Applications-supported client and is installed along with Oracle Application Object Library.
- The E-mail notifications component contains a program that can send mail through Oracle Office/InterOffice, UNIX Sendmail, or a Windows NT MAPI-compliant mail application. Oracle Workflow can also send mail to other E-mail applications as long as you install the appropriate Oracle Office/InterOffice or UNIX gateway product to communicate with your E-mail application of choice.
- To send and respond to E-mail notifications with HTML attachments, your E-mail application should support HTML

attachments and you should have a Web browser application that supports JavaScript and Frames to view the attachment.

- The Web notifications and Workflow Monitor components require Oracle WebServer to be installed first. To view notifications you need a Web browser application that supports JavaScript and Frames. To view the Workflow Monitor you need a Web browser that supports Java Development Kit (JDK), Version 1.1.4 and Abstract Windowing Toolkit (AWT).

Implementing Oracle Workflow

After you install Oracle Workflow, you implement it for your site by setting up the roles, icons, notification templates, background engines, and access levels appropriate for your enterprise.

Required Set Up Steps

- ❑ Step 1: If you are using the standalone version of Oracle Workflow, you must map Oracle Workflow's directory service to the users and roles currently defined in your organization's directory repository by constructing views based on those database tables. The Notification System uses these views to send notifications to the performers specified in your activities. Your roles can be either individual users or a group of users. Oracle Workflow provides example directory services views that you can modify and reload. See: [Setting Up an Oracle Workflow Directory Service: page 2 – 7](#).
- ❑ Step 2: If you are using the standalone version of Oracle Workflow, you must create a view called WF_LANGUAGES that identifies the languages defined in your Oracle7 installation. Oracle Workflow uses this view to create in its translation tables, a row that maps to a row found in its non-translated base table for each installed language. See: [Creating the WF_LANGUAGES View: page 2 – 13](#).
- ❑ Step 3: If you are using the standalone version of Oracle Workflow, you must define an environment variable called WF_RESOURCES. See: [Setting the WF_RESOURCES Environment Variable: page 2 – 14](#).
- ❑ Step 4: After you install and configure Oracle WebServer, identify the Web Agent to be used by Oracle Workflow. See: [Identifying the Oracle Web Agent used by Oracle Workflow: page 2 – 15](#).
- ❑ Step 5: Once you define your Oracle Workflow directory service, you need to identify the role that should have access to Oracle Workflow's administration features such as the Find Processes web page. See: [Identifying the Oracle Workflow Administration Role: page 2 – 18](#).
- ❑ Step 6: Configure Oracle WebServer to point to the Java code required to run the Workflow Monitor and optionally customize the company logo that appears in Oracle Workflow's web pages. See: [Setting Up the Workflow Monitor and Oracle Workflow's Web Pages: page 2 – 19](#).

- ❑ Step 7: If you are using the standalone version of Oracle Workflow, secure your Workflow database connection using Oracle WebServer security. See: [Secure the Workflow Database Connection Descriptor](#): page 2 – 21.
- ❑ Step 8: Set up the Notification Mailer program so that users can receive notifications by E-mail if that is an option you are providing to your users. See: [Implementing the Notification Mailer](#): page 2 – 25.
- ❑ Step 9: Set up background Workflow Engines to control the load and throughput of the primary Workflow Engine on your system. You can specify the cost threshold level of your primary and background engines to determine the activities an engine processes and the activities an engine defers. See: [Setting Up Background Workflow Engines](#): page 2 – 43.

Optional Set Up Steps

- ❑ Step 10: You can modify the templates for your electronic mail notifications. See: [Modifying Your Message Templates](#): page 2 – 36.
- ❑ Step 11: You can include additional icons to your Oracle Workflow Icons subdirectory to customize the diagrammatic representation of your workflow processes. Use custom icons to provide meaningful symbols for each activity you define. See: [Adding Custom Icons to Oracle Workflow](#): page 2 – 47.

Other Workflow Features

Before deploying Oracle Workflow and custom process definitions to other branches of your enterprise, you can protect your data from further modification by determining the level of access your users have to the data. See: [Overview of Oracle Workflow Access Protection](#): page 2 – 48.

You can also use the Workflow Definitions Loader to load workflow process definitions from flat files to the database without using Oracle Workflow Builder. See: [Using the Workflow Definitions Loader](#): page 2 – 54.

Identifying the Version of Your Oracle Workflow Server

If you ever need to determine the version of the Oracle Workflow server you are running, you can connect to your Workflow server account using SQL*PLUS and run a script called wfver.sql. See: wfver.sql: page 11 – 7.

Setting Up an Oracle Workflow Directory Service

Oracle Workflow offers you flexibility in defining who your workflow users and roles are. You determine the directory repository you want Oracle Workflow to reference for users and roles information by creating three views based on the database tables that make up that repository. The views are: WF_USERS, WF_ROLES, and WF_USER_ROLES.

In addition, Oracle Workflow provides three local tables called WF_LOCAL_USERS, WF_LOCAL_ROLES, and WF_LOCAL_USER_ROLES which you can use to add information about users and roles not included in your existing directory repository.

Note: Currently you must use SQL*PLUS or create your own custom application interface to enter data into these WF_LOCAL tables.

WF_USERS

The WF_USERS view should reference information about all the individuals in your organization who may receive workflow notifications. Create this view, making sure it contains the following columns:

- **Name**—The internal name of the user as referenced by the Workflow Engine and Notification System. For example, an internal name for a user can be mbeech or 009, where 009 represents the user's employee ID.



- Attention:** The Name column must be sourced from a column that is less than 30 characters long and is all uppercase. If your source table does not have a column that meets these criteria, DO NOT use string functions to force these restrictions. Instead, define the Name column to be <orig_system>:<orig_system_id> so that Oracle Workflow can reference the original base table where users are stored and a unique user in that table. For example, "PER_PEOPLE:009" represents a user whose employee ID is 009 and is stored in the personnel table called PER_PEOPLE.
- **Display_Name**—The display name of the user. An example of a display name can be 'Beech, Matthew'.
 - **Description**—An optional description of the user.
 - **Notification_Preference**—Indicate how this user prefers to receive notifications. A value of MAILTEXT or MAILHTML allows users to receive and respond to notifications by E-mail or by

E-mail with HTML attachments, respectively. A value of `QUERY` allows users to query notifications from the Notifications Web page or Notification Viewer form. Finally, a value of `SUMMARY` allows users to get periodic E-mail summaries of their open notifications. However, to respond to the individual notifications, they have to query the notification from the Notification Web page or Notification Viewer form. See: Overview of Notification Handling; page 8 – 2.

Note: A notification preference of `MAILTEXT` or `MAILHTML` also allows users to query their notifications from the Notifications Web page or Notification Viewer form.

- **Language**—The value of the Oracle7 `NLS_LANGUAGE` initialization parameter that specifies the default language-dependent behavior of the user’s notification session. Refer to your Oracle7 user’s guide or installation manual for the list of supported language conventions.
- **Territory**—The value of the Oracle7 `NLS_TERRITORY` initialization parameter that specifies the default territory-dependant date and numeric formatting used in the user’s notification session. Refer to your Oracle7 user’s guide or installation manual for the list of supported territory conventions.
- **Email_Address**—A valid electronic mail address for this user or a mail distribution list defined by your electronic mail system.
- **Fax**—A Fax number for the user.
- **Orig_System**—A code that you assign to the directory repository that this view is based on. For example, if this view is based on the personnel data stored in a Human Resource Management System, `Orig_System` can be defined as `PER`.
- **Orig_System_ID**—The primary key that identifies the user in this repository system. For example, `Orig_System_ID` can be defined as the value stored in a column called `PERSON_ID` in a Human Resources database table called `PER_PEOPLE`.
- **Status**—The availability of the user to participate in a workflow process. The possible statuses are: active (`ACTIVE`), unavailable for an extended period (`EXTLEAVE`), permanently unavailable (`INACTIVE`), and temporarily unavailable (`TMPLEAVE`). These statuses are also stored in the lookup type called `WFSTD_AVAILABILITY_STATUS`.

WF_ROLES

The WF_ROLES view should reference information about all the roles in your organization who may receive workflow notifications. Create this view, making sure it contains the following columns pertaining to the roles in your repository, similar to those described for WF_USERS:



Attention: We require that you also define each user identified by WF_USERS as a role.

Note: If a user is a member of a role and the user information is different from the role information, the role information will override the user information when the Notification System delivers a notification to the role. For example, suppose a user has a notification preference of 'SUMMARY', and the user is also a member of a multi-user role, whose notification preference is 'MAILHTML'. When a notification is assigned to the multi-user role, the user will receive a single notification message addressed to the role, as opposed to a summary message that includes that notification in it.

- *Name
- *Display_Name
- Description
- *Notification_Preference
- *Language
- *Territory
- *Email_Address
- *Fax
- Orig_System
- Orig_System_ID



Attention: The Name column must be sourced from a column that is less than 30 characters long and is all uppercase. If your source table does not have a column that meets these criteria, DO NOT use string functions to force these restrictions. Instead, define the Name column to be <orig_system>:<orig_system_id> so that Oracle Workflow can reference the original base table where roles are stored and a unique role in that table. For example, "PER_POSITION:009" represents a position whose ID is 009 and is stored in the personnel table called PER_POSITION.

WF_USER_ROLES

The WF_USER_ROLES view is an intersection of the users and roles in WF_USERS and WF_ROLES. Create this view, making sure it contains the following columns:

- User_Name—The internal name of the user as listed in the view WF_USERS.
- User_Orig_System—A code that you assign to the user directory repository as listed in the view WF_USERS.
- User_Orig_System_ID—The primary key that identifies the user in the user directory repository as listed in the view WF_USERS.
- Role_Name—The internal name of the role as listed in the view WF_ROLES.
- Role_Orig_System—A code that you assign to the role directory repository as listed in the view WF_ROLES.
- Role_Orig_System_ID—The primary key that identifies the role in the role directory repository as listed in the view WF_ROLES.



Attention: To take advantage of unique indexes when querying users, make sure you initially enter the usernames in your database in uppercase only. Forcing the usernames to uppercase in your view definition results in poor performance when accessing these views.



Warning: Avoid making a join to a view that contains a union as this results in poor database performance. Oracle7 is currently unable to preserve the indexes in that view when you make such a join. The workflow directory services views you create will most likely contain unions, therefore you should not join to them directly. If you need to retrieve data from any of the three directory services views, use the appropriate directory services API. See: Workflow Directory Services APIs: page 7 – 49.

Predefined Directory Services

Oracle Workflow provides scripts for you to implement any one of three directory service environments. If you are using the version of Oracle Workflow embedded in Oracle Applications you automatically:

- Integrate your Oracle Workflow directory service with a unified Oracle Applications environment.

If you are using the standalone version Oracle Workflow, you can choose to implement one of the following two directory services or create your own:

- A directory services with native Oracle users.
- A directory services with local workflow users.

You can customize any of these directory services environments further by editing and rerunning their scripts against your Workflow Server.



Attention: If you create your own directory service or edit any of the predefined directory services listed above, you should run the script *wfdirchk.sql* to validate your directory service data model. The script is located on your server in the Oracle Workflow *admin/sql* subdirectory for the standalone version of Oracle Workflow, or in the *sql* subdirectory under *\$FND_TOP* for the version of Oracle Workflow embedded in Oracle Applications. See: *Wfdirchk.sql*: page 11 – 7.

Integrating Oracle Workflow Directory Services with a Unified Oracle Applications Environment

If you are using the version of Oracle Workflow embedded in Oracle Applications, your Oracle Workflow directory service views are automatically based on a unified Oracle Applications environment. The unified environment includes joins to tables in Oracle Human Resources, Oracle Application Object Library, and various other Oracle Applications, as well as to the *WF_LOCAL* tables.

Oracle Workflow provides a sql script that defines the *WF_USERS*, *WF_ROLES*, and *WF_USER_ROLES* views to map to this unified environment. When you install Oracle Applications, you automatically install this script to create the unified environment. However, if you should need to edit and rerun this script for whatever reason, the script is called *wfdirhrv.sql* and is located on your server in the *sql* subdirectory under *\$FND_TOP*.

Aside from the users and roles stored in *WF_LOCAL_USERS* and *WF_LOCAL_ROLES*, the default notification preference for all workflow users and roles in the unified environment is set to 'MAILHTML'.

Integrating Oracle Workflow Directory Services with Native Oracle Users

If you plan to use the standalone version of Oracle Workflow, you can map your directory service to the native users and roles in the Oracle

RDMBS. You base your views on the tables `DBA_USERS` and `DBA_ROLES`.

Oracle Workflow provides a script you can use to setup the views. Use the *wfdirouv.sql* script in the Oracle Workflow *sql* subdirectory on your server. This script creates three views.

The `WF_USERS` view creates a workflow user for each DBA user. The originating system is called `ORACLE`, and the `USERNAME` column in `DBA_USERS` is the originating system ID. The default notification preference is `MAILHTML`.

The `WF_ROLES` view includes all users in the `WF_USERS` view and all roles defined in the `DBA_ROLES` table as workflow roles. The originating system is `ORACLE` and the originating system ID is the `ROLE` column in `DBA_ROLES`.

The `WF_USER_ROLES` view consists of the names and originating system information of both users and roles in `WF_USERS` and `WF_ROLES`.

Integrating Oracle Workflow Directory Services with Local Workflow Users

If you plan to use the standalone version of Oracle Workflow, you can map your directory service to the local workflow users and roles stored in the `WF_LOCAL_USERS` and `WF_LOCAL_ROLES` tables.

Oracle Workflow provides a script you can use to setup the views. Use the *wfdircsv.sql* script in the Oracle Workflow *sql* subdirectory on your server. This script creates three views. You can customize the views in this script to incorporate the users and roles from you custom directory repository.

The originating system in the `WF_USERS` view is called `WF_LOCAL`, and the originating system ID is 0.

The `WF_ROLES` view includes all users in `WF_LOCAL_USERS` and all roles defined in `WF_LOCAL_ROLES`. The originating system is `WF_LOCAL` and the originating system ID is 0.

The `WF_USER_ROLES` view consists of the names and originating system information of both users and roles in `WF_USERS` and `WF_ROLES`.

Creating the WF_LANGUAGES View

The field values in the property pages of Oracle Workflow Builder and the workflow notifications delivered to your users can be translated to the languages defined in your Oracle installation. However, in order for this to be possible, you must create a view called WF_LANGUAGES that identifies the languages defined in your Oracle installation. Oracle Workflow uses this view to create in its translatable tables, a row for each language that maps to a row found in its non-translated base table.

The WF_LANGUAGES view must include the following columns:

- Code—The language code.
- Display_Name—The display name of the language.
- NLS_Language—The value of the Oracle NLS_LANGUAGE initialization parameter that specifies the default language-dependent behavior of a session.
- NLS_Territory—The value of the Oracle NLS_TERRITORY initialization parameter that specifies the default territory-dependant date and numeric formatting of a session.
- NLS_Codeset—The code set for the language.
- Installed_Flag—Flag to indicate if the language is installed and available for use.

A sample WF_LANGUAGES view is included in the script of each of the predefined directory services that Oracle Workflow provides.

Setting the WF_RESOURCES Environment Variable

If you are using the standalone version of Oracle Workflow, you must set an environment variable called WF_RESOURCES to point to the language-dependent Oracle Workflow resource file (wf<language>.res). The resource file generally resides under the *res* subdirectory of your Oracle Workflow server directory structure.

You do not need to set this environment variable if you are using the version of Oracle Workflow embedded in Oracle Applications. For Oracle Applications, the path of the language-dependent Oracle Workflow resource file is \$FND_TOP/\$APPLRSC/wf<language>.res.

Identifying the Oracle Web Agent used by Oracle Workflow

Oracle WebServer must be installed before installing Oracle Workflow. Once you finish installing and configuring Oracle WebServer, you must identify the Oracle Web Agent that Oracle Workflow should use to access its Web components.

► **To Identify a Web Agent for Oracle Workflow**

1. Edit the file `wcfg.msg` as follows:

```
WFTKN WF_WEB_AGENT                                0 <your_web_agent_here>
```

Replace `<your_web_agent_here>` with the base URL of the Oracle Web Agent you defined for Oracle Workflow in Oracle WebServer. The base URL should look similar to this:

```
http://<server.com:portID>/<PLSQL_agent_virtual_path>
```

`<server.com:portID>` represents the server and TCP/IP port number on which your Web Listener accepts requests and `<PLSQL_agent_virtual_path>` represents the application virtual path of your Oracle Workflow PL/SQL agent. Each PL/SQL agent you configure connects to a particular database schema. See your Oracle WebServer documentation for more information.

Note: If you are using the version of Oracle Workflow embedded in Oracle Applications, the file `wcfg.msg` is located in the `resource/<language>` subdirectory under `$FND_TOP`. If you are using the standalone version of Oracle Workflow, the file is located in the Oracle Workflow server `res/<language>` subdirectory.

2. Run the Workflow Resource Generator to load the contents of `wcfg.msg` into the table `WF_RESOURCES`. See: `To run the Workflow Resource Generator: 2 – 15`

► **To run the Workflow Resource Generator:**

For the standalone version of Oracle Workflow:

1. The Workflow Resource Generator program is located in the `bin` subdirectory of the Oracle Workflow directory structure.
2. Run the program from your operating system prompt as follows:
 - To generate a binary resource file from a source file (`.msg`), type:

```
wfresgen [-v] -f <resourcefile> <source_file>
```

Replace *<resourcefile>* with the full path and name of the resource file you want to generate, and *<source_file>* with the full path and name of your source file. The optional `-v` flag causes the program to validate the source file against the binary resource file.

- To upload seed data from a source file (.msg) to the database table WF_RESOURCES, type:

```
wfresgen [-v] -d <username/password@database>  
<source_file>
```

Replace *<username/password@database>* with the username, password and SQL*Net connect string or alias to your database and *<source_file>* with the full path and name of the source file you want to upload. The optional `-v` flag causes the program to validate the source file against the database.

For Oracle Workflow embedded in Oracle Applications:

1. The Workflow Resource Generator program is registered as a concurrent program. You can run the Workflow Resource Generator concurrent program from the Submit Requests form or from the command line.
2. To run the concurrent program from the Submit Requests form, navigate to the Submit Requests form.

Note: Your system administrator needs to add this concurrent program to a request security group for the responsibility that you want to run this program from. See: Overview of Concurrent Programs and Requests, *Oracle Applications System Administrator's Guide*
3. Submit the Workflow Resource Generator concurrent program as a request. See: Submitting a Request, *Oracle Applications User's Guide*.
4. In the Parameters window, enter values for the following parameters:

Destination Type Specify "Database", to upload seed data to the database table WF_RESOURCES from a source file (.msg), or "File", to generate a resource file from a source file.

Destination If you specify "File" for Destination Type, then enter the full path and name of the resource file you wish to generate. If you specify "Database" for Destination Type, then the program automatically uses the current database account as its destination.

Source Specify the full path and name of your source file.

5. Choose OK to close the Parameters window.
6. When you finish modifying the print and run options for this request, choose Submit to submit the request.
7. Rather than use the Submit Requests form, you can also run the Workflow Resource Generator concurrent program from the command line using one of two commands. To generate a resource file from a source file, type:

```
WFRESGEN apps/pwd 0 Y FILE res_file source_file
```

To upload seed data to the database table WF_RESOURCES from a source file, type:

```
WFRESGEN apps/pwd 0 Y DATABASE source_file
```

Replace *apps/pwd* with the username and password to the APPS schema, replace *res_file* with the file specification of a resource file, and replace *source_file* with the file specification of a source file (.msg). A file specification is specified as:

```
@<application_short_name>:[<dir>/.../]file.ext
```

or

```
<native path>
```

Identifying the Oracle Workflow Administration Role

You can specify who has administrator privileges in Oracle Workflow by defining an Oracle Workflow administration role. Any user in the administration role can run the Oracle Workflow Find Processes web page, which provides full access to Oracle Workflow's administration features. In addition, any user in the administration role can view any other user's notifications. See: *Setting Up an Oracle Workflow Directory Service*: page 2 – 7.

► **To define the Oracle Workflow administration role**

1. Edit the file `wfcfg.msg` as follows:

```
WFTKN WF_ADMIN_ROLE                0 <role_name>
```

Replace `<role_name>` with the internal name of a role defined in the Oracle Workflow directory service. For example:

```
WFTKN WF_ADMIN_ROLE                0 SYSADMIN
```

Any user associated with this role will have full workflow administration privileges.

Note: If you are using the version of Oracle Workflow embedded in Oracle Applications, the file `wfcfg.msg` is located in the `resource/<language>` subdirectory under `$FND_TOP`. If you are using the standalone version of Oracle Workflow, the file is located in the Oracle Workflow server `res/<language>` subdirectory.

2. If you want all users and roles to have workflow administration privileges, such as in a development environment, replace `<role_name>` with an asterisk (*) as follows:

```
WFTKN WF_ADMIN_ROLE                0 *
```

3. Run the Workflow Resource Generator to load the contents of `wfcfg.msg` into the table `WF_RESOURCES`. See: *To run the Workflow Resource Generator*: 2 – 15.

Setting Up the Workflow Monitor and Oracle Workflow's Web Pages

To use Oracle Workflow's web pages and the Workflow Monitor at your site, you must first install Oracle WebServer. Refer to your Oracle WebServer documentation for additional information.

Once Oracle WebServer is installed, you can customize the company logo that appears on Oracle Workflow's web pages. In addition, you must configure Oracle WebServer to point to the Java code required to run the Workflow Monitor.

Use a web browser that supports JavaScript to connect to the Notification Web page or a web browser that supports Java Development Kit (JDK), Version 1.1.4 and Abstract Windowing Toolkit (AWT) to connect to the Workflow Monitor.

► **To Set Up Oracle Workflow's Web Components:**

1. Copy or rename your company logo file (in .gif format) to **WFLOGO.gif** located in the appropriate directory.

For the standalone version of Oracle Workflow:

```
<ORACLE_HOME>/wf/java/oracle/wf/
```

```
<ORACLE_HOME>/wf/java represents your <java_directory_path>.
```

For the Oracle Applications–embedded version of Oracle Workflow:

```
<java_directory_path>/oracle/wf/
```

2. Connect to the Oracle WebServer home page. The URL to use is specific to your installation, but is typically a specific port on the server machine where you installed Oracle WebServer:

```
http://<servername>:<portID>/
```

If you are using Oracle WebServer 2.x, complete Steps 3 through 8. If you are using Oracle WebServer 3.x, complete Steps 9 through 15.

3. Choose WebServer Manager to go to the Oracle WebServer Administration page.
4. Choose Oracle Web Listener from the Oracle WebServer Administration page to go to the Oracle Web Listener Administration page. Enter the username and password for your Admin Server.

5. Scroll towards the bottom of the page to the Oracle Web Listeners list and locate the listener that you configured for the Oracle Workflow Server. Choose CONFIGURE for that Listener.
6. In the Oracle Web Listener Administration Server Advanced Configuration page, scroll to the Oracle Web Listener Configuration Parameters section and choose Directory Mappings to go to the Directory Mappings section.
7. Add the following entry in the Directory Mappings section to map the directory structure where **java** is located to a directory structure where the Java code exists:

<u>File-System Directory</u>	<u>Flag</u>	<u>Virtual Directory</u>
<i>/<java_directory path>/</i>	NR	<i>/wfjava/</i>

8. Choose Modify Listener.
9. Enter the username and password for your Oracle Web Application Server.
10. Choose Web Application Server Manager to go to the Administration home page.
11. Choose Oracle Web Listener from the Administration home page to go to the Oracle Web Listener Administration page.
12. Scroll towards the bottom of the page to the Oracle Web Listeners list and locate the listener that you configured for the Oracle Workflow Server. Choose CONFIGURE for that Listener.
13. In the Oracle Web Listener Advanced Configuration page, choose the Directory link from the list of links on the left hand frame to go to the Directory Mappings section.
14. Add the following entry in the Directory Mappings section to map the directory structure where **java** is located to a directory structure where the Java code exists:

<u>File-System Directory</u>	<u>Flag</u>	<u>Virtual Directory</u>
<i>/<java_directory path>/</i>	NR	<i>/wfjava/</i>

15. Choose Modify Listener.

See Also

Viewing Notifications from a Web Browser: page 8 – 12

Workflow Monitor: page 9 – 3

Secure the Workflow Database Connection Descriptor (DCD)

If you are not using Oracle Self-Service Web Applications with Oracle Workflow, then Oracle Workflow's web pages must rely on the user authentication feature of Oracle WebServer to provide security. To ensure that only authorized users can access workflow processes, the URLs that generate Oracle Workflow's web pages must be protected by the Oracle WebServer authentication feature. This feature requires users to enter a username and password before accessing these pages by protecting the Oracle Workflow database connection descriptor (DCD). Follow the steps listed:

- To protect the DCD in Oracle WebServer 2.x

or

- To protect the DAD in Oracle WebServer 3.x

Note: The database access descriptor (DAD) in Oracle WebServer 3.x is synonymous to the database connection descriptor (DCD) in Oracle WebServer 2.x.

► To protect the DCD in Oracle WebServer 2.x:

1. Connect to the Oracle WebServer Administration page.
2. Choose the Oracle Web Listener link.
3. Choose Configure for the appropriate listener.
4. Choose Security: Access Control and Encryption.
5. Enter usernames and passwords in either the Basic or Digest Authentication sections.

Basic authentication allows you to assign passwords to users, assign users to groups, and define sets of users and groups, called "realms." You can then assign the users, groups, and realms to specific files and directories, requiring requestors to provide a username and password to gain access. Basic authentication sends unencrypted passwords across the network, making this method subject to subversion. Basic authentication is not recommended when security is critical.

Digest authentication is the same as basic authentication except that it sends passwords encrypted across the network in the form of a cryptographic checksum, also called a "digest." You should use this scheme whenever authentication is required, although some older web browsers may not support it.

6. Assign your users to a group for the appropriate authentication method.
7. Assign the group to a realm for the appropriate authentication method.
8. Choose the Modify Listener button to save your changes.
9. Navigate back to the Listener Administration page.
10. Choose Web Request Broker and choose the Modify link.
11. Scroll to the Protecting Applications section.
12. Enter the following values in the fields:

Virtual Path	Scheme	Realm
<code><virtual_path></code>	<code><Basic/Digest/></code>	<code><realm_name></code>

`<virtual_path>` represents the virtual path of the PL/SQL agent's shared files, as defined in the Applications and Directories section of the Web Request Broker Administration page. Specify the scheme as either Basic or Digest. `<realm_name>` represents the realm name that you specified for your authentication scheme.

13. Choose Modify WRB Configuration to save your changes.
14. Restart the listener.

Note: You must set protection in the Web Request Broker section and not the Protection section of the Listener Administration page.

► **To protect the DAD in Oracle WebServer 3.x:**

1. Connect to the Oracle Web Application Server Administration page.
2. Choose the Oracle Web Application Server link.
3. Choose the Authorization Server link.
4. Select either the Basic, Digest, or Database authentication scheme by choosing the appropriate link.

Basic authentication allows you to assign passwords to users, assign users to groups, and define sets of users and groups, called "realms." You can then assign the users, groups, and realms to specific files and directories, requiring requestors to provide a username and password to gain access. Basic authentication sends unencrypted passwords across the network, making this method subject to subversion. Basic authentication is not recommended when security is critical.

Digest authentication is the same as basic authentication except that it sends passwords encrypted across the network in the form of a cryptographic checksum, also called a "digest." You should use this scheme whenever authentication is required, although some older web browsers may not support it.

Database authentication allows you to authenticate the username and password pair against a database by using the username and password to logon to an Oracle RDBMS. The realm of database authentication consists of two parts: a Database Access Descriptor (DAD) and optionally a database role. The DAD identifies the database to check against. The username and password, if available in the DAD, is ignored. The database role allows that only a subset of database users (those who have the privilege to assume the role) be authenticated.

5. If you select either Basic or Digest authentication, enter usernames and passwords for your users, assign your users to a group, then assign the group to a realm for your authentication method.

If you select Database authentication, assign groups to a realm, then for each group, specify the DAD to check against, and optionally specify the roles to be authenticated.

6. Choose Modify to save your changes.
7. Navigate back to the Oracle Web Application Server Administration page.
8. Choose Cartridge Administration, then Cartridge Summary (Web Request Broker).
9. Choose the Protection link in the frame on the left side of the page to go to the Protecting Applications section.
10. Enter the following values in these fields to protect your realm:

<u>Virtual Path</u>	<u>Scheme</u>	<u>Realm</u>
<i><virtual_path></i>	<i><Basic/Digest/ Basic_Oracle></i>	<i><realm_name></i>

<virtual_path> represents the virtual path of the PL/SQL cartridge's shared files, as defined in the Applications and Directories section of the Web Request Broker Administration page. Specify the scheme as either Basic, Digest, or Basic_Oracle (for the Database scheme). *<realm_name>* represents the realm name that you specified in Step 5.

11. Choose Modify WRB Configuration to save your changes.
12. Restart the listener.

See Also

Viewing Notifications from a Web Browser: page 8 – 12

Implementing the Notification Mailer

The Notification Mailer is a program that performs E-mail send and response processing for the Oracle Workflow Notification System. It polls the database for messages that have to be sent, and performs the following action for each message:

- Resolves the recipient role to a single E-mail address, which itself can be a mail list.
- Switches its database session to the role's preferred language and territory as defined by the directory service.
- Generates the message and any optional attachments using the appropriate message template.
- Sends the message via UNIX Sendmail, Oracle Office/InterOffice, or any MAPI-compliant mail application on Windows NT.

The Notification Mailer also processes responses by interpreting the text of messages mailed to its response mail account and calling the appropriate notification response function to complete the notification.

Once you set up the Notification Mailer to run, it continually polls the database for messages to send and checks its response mail account for responses to process. You do not have to do anything else unless you have a need to shut it down and restart it again with different parameters.



Attention: The Notification Mailer will shut itself down if a database failure is encountered or if the PL/SQL package state for the session is invalid due to dropping or replacing of package definitions. If you are using the standalone version of Oracle Workflow, you can restart the Notification Mailer manually or run a shell script that restarts the Notification Mailer if it ever exits with a failure. See: *To Run a Perpetual Shell Script for the Notification Mailer: page 2 – 33*. If you are using the version of Oracle Workflow embedded in Oracle Applications, you can use the concurrent manager to restart the Notification Mailer program manually or schedule it to restart periodically.

You can install and set up the Notification Mailer to run against UNIX Sendmail, Oracle Office/InterOffice, or a MAPI-compliant mail application on Windows NT. However, before doing so, you must set up a least one mail account for the Notification Mailer in one of these three mail applications. You must also define three folders or files in your mail account to use response processing.

See Also

Reviewing Notifications via Electronic Mail: page 8 – 6

Starting the Notification Mailer

- ▶ **To start the Notification Mailer for UNIX Sendmail or Oracle Office/InterOffice:**

For the standalone version of Oracle Workflow:

1. The Notification Mailer resides on your server in the *bin* subdirectory of your Oracle Workflow directory structure. Type the following command at your operating system prompt:

```
wfmail.<xxx> -f <config_file>
```

Replace *<xxx>* with *ofc* to use the Oracle Office/InterOffice version of the Notification Mailer or with *snd* to use the UNIX Sendmail version. Replace *<config_file>* with the full path and name of the configuration file that contains the parameters you want to run with the Notification Mailer.

2. Alternatively, you can specify the parameters for the Notification Mailer as arguments on the command line rather than in a configuration file, by typing the following command:

```
wfmail.<xxx> <arg1> <arg2> ...
```

Or, you can specify a configuration file, but override certain parameter values in the configuration file by specifying command line values:

```
wfmail.<xxx> -f <config_file> <arg1> <arg2> ...
```

Replace *<arg1> <arg2> ...* with any number of optional parameters and values, using the format `parameter=value`.

For the version of Oracle Workflow embedded in Oracle Applications:

1. The Notification Mailer program is registered as a concurrent program. You can run the Notification Mailer concurrent program from the Submit Requests form or from the command line.
2. To run the concurrent program from the Submit Requests form, navigate to the Submit Requests form.

Note: Your system administrator needs to add this concurrent program to a request security group for the responsibility that you want to run this program from. See: Overview of Concurrent Programs and Requests, *Oracle Applications System Administrator's Guide*

3. Submit the Notification Mailer concurrent program as a request. See: Submitting a Request, *Oracle Applications User's Guide*.
4. In the Parameters window, enter the path and filename of a configuration file. The configuration file contains the parameters you want to run with the Notification Mailer.
5. Choose OK to close the Parameters window.
6. When you finish modifying the print and run options for this request, choose Submit to submit the request.
7. Rather than use the Submit Requests form, you can also run the Notification Mailer concurrent program from the command line. Enter:

```
WFMAIL apps/pwd 0 Y FILE config_file
```

Replace *apps/pwd* with username and password to the APPS schema, replace *config_file* with the file specification of the configuration file that contains the parameters you want to run with the Notification Mailer.

A file specification is specified as:

```
@<application_short_name>:[<dir>/.../]file.ext
```

or

```
<native path>
```

► **To start the Notification Mailer for MAPI-compliant Mail Applications:**

1. Install the Notification Mailer for MAPI-compliant mail applications on your Windows NT PC using Oracle Installer. The Notification Mailer program resides in
`<drive>:\<ORACLE_HOME>\wf20\bin.`
2. Start the Notification Mailer program by entering the following command in an MS-DOS prompt window:

```
<drive>:\<ORACLE_HOME>\wf20\bin\wfmlr20.exe -f  
<config_file>
```


Replace *<config_file>* with the full path and name of the configuration file that contains the parameters you want to run with the Notification Mailer.

Note: You can also double-click on the Oracle Workflow Notification Mailer icon in the Oracle for Windows NT program group to start the program, but you must first edit the properties of the icon to include the above command as its target.

3. Alternatively, if you want to specify the parameters for the Notification Mailer as arguments on the command line rather than in a configuration file, you can type the following command:

```
wfmlr20.exe <arg1> <arg2> ...
```

Or, you can specify a configuration file, but override certain parameter values in the configuration file by specifying command line values:

```
wfmlr20.exe -f <config_file> <arg1> <arg2> ...
```

Replace *<arg1> <arg2> ...* with the required and optional parameters and values, using the format `parameter=value`.

► **To create a configuration file for the Notification Mailer:**

1. Oracle Workflow provides an example configuration file, called *wfmail.cfg*. If you are using the standalone version of Oracle Workflow, the file resides in your Oracle Workflow server directory structure, under the subdirectory *res*. For the version of Oracle Workflow embedded in Oracle Applications, the file resides in the *resource* subdirectory under *\$FND_TOP* on your server. The file also resides on your PC in the

<drive>:\<ORACLE_HOME>\wf20\data subdirectory.

2. The content of the configuration file is formatted as follows:


```
#Description  
PARAMETER1=<value1>
```

```
#Description  
PARAMETER2=<value2>
```

...

Any text preceded by # is not interpreted and can be used for including comments. List each parameter name on the left side of the equal sign (=) and specify a value for each parameter on the right.

3. The parameters are as follows:

- CONNECT** (Required) The information to connect to the database account where the Oracle Workflow server is installed, using the format, *username/password@connect_string* (or *alias*).
- ACCOUNT** (Required) The information to connect to the mail account that the program uses to send notification messages. For MAPI-compliant mail programs, the account information is the mail account profile name and mail account password. For Oracle Office/InterOffice, the account information would be an Oracle Office/InterOffice database account of the format, *username/password@connect_string* (or *alias*). For Sendmail, the account information would be the full path of the outgoing mail spool account file, which also corresponds to the account from which you start the Notification Mailer.
-  **Attention:** If you are using the version of Oracle Workflow embedded in Oracle Applications, and want to start the Sendmail version of the Notification Mailer concurrent program, then the Account parameter must be set to the account from which you start the Concurrent Manager.
- NODE** (Required) The node identifier name. You can have multiple workflow databases route messages through the same mail account. By defining an identifying node name for each Notification Mailer running against each database, responses can be routed back to the correct database without requiring database connection information to be included in the message. The node name is included with the outgoing notification ID. The default name is `main`.
- FROM** The value that appears in the From: field of the message header when a notification message is delivered to a user.
- SUMMARYONLY** (Required) Indicate whether this Notification Mailer processes only notifications assigned to users/roles with a notification preference of 'SUMMARY' or whether it only processes notifications for users/roles with a notification

preference of 'MAILTEXT' or 'MAILHTML'. Valid values are Y or N. The default is N. You should set up at least two Notification Mailers, one where SUMMARYONLY=Y and one where SUMMARYONLY=N if any of your workflow users or roles have a notification preference of 'MAILTEXT', 'MAILHTML', or 'SUMMARY'. See: Setting Up Users and Roles from a Directory Repository: 2 – 7.



Attention: If you set SUMMARYONLY=Y, then the Notification Mailer will shut itself down after it polls the database and delivers any appropriate notification summaries. You must therefore schedule the Notification Mailer to run at the frequency you want notification summaries to be delivered. We recommend you run the summary Notification Mailer once a day, since the summary includes all open notifications. For Oracle Workflow running in the standalone environment, this would involve creating a operating system script, such as a cron job in UNIX, to schedule the Notification Mailer. For the version of Oracle Workflow embedded in Oracle Applications, this simply involves scheduling the Notification Mailer concurrent program in the Submit Request form.

- IDLE** The number of seconds to wait before checking for messages to send. The value must be an integer greater than or equal to zero. The default is 60 .
- LOG** The name of a log file to record activity. A valid value would be a filename. This parameter is valid only for the standalone version of the Notification Mailer. For the concurrent program version of the Notification Mailer, the activity output goes to the concurrent manager log file.
- SHUTDOWN** The name of a file that cues the Notification Mailer to shut down. This lets you safely shut down the Notification Mailer without killing the process. The Notification Mailer always looks for the shutdown file in its current working directory before looking for notifications to process. If the file exists, then the Notification Mailer shuts down. You must remove the shutdown file to restart the Notification Mailer again. The default filename is shutdown.

For the standalone version of Oracle Workflow, the Notification Mailer's current working directory is the directory from which you start the Notification Mailer. For the version of Oracle Workflow embedded in Oracle Applications, the current working directory is the \$APPLCSF/\$APPLLOG directory. If you have not set the \$APPLCSF environment variable, then place the shutdown file in the \$FND_TOP/\$APPLLOG directory.

FAILCOMMAND	The command to run if the Notification Mailer encounters a fatal error.
DEBUG	Indicate whether to print debugging information in the log. Valid values include Y or N. The default is N.
TEST_ADDRESS	Indicate a test E-mail address to direct all outgoing E-mail notifications. The test address overrides each recipient's E-mail address so that you can test a workflow process without having to change each recipient's E-mail address to access the test notifications.
REPLYTO	A default E-mail address to reply to, if the E-mail account that processes responses is different from the E-mail account that sends outgoing notifications.
HTMLAGENT	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 token WF_WEB_AGENT stored in the WF_RESOURCES table, but you can override this default by entering a different value for this parameter. See: Identifying the Oracle Web Agent used by Oracle Workflow: page 2 – 15.
HTMLFILE	The filename of the default attachment. The default is <code>attach.html</code> .
HTMLDESC	A description of the default attachment. The default is <code>HTML</code> .
HTMLTYPE	The html attachment type number. This argument is required only if you are using Oracle Office/InterOffice and want to include an HTML

attachment with your E-mail notification. The default is 10003.

DISCARD

The name of the mail folder or full path name of the mail file to put discarded messages. A '-' preceding the name causes the Notification Mailer to truncate the folder or file on startup. The default is `discard`.

PROCESS

The name of the mail folder or full path name of the mail file to put processed notification messages. A '-' preceding the name causes the Notification Mailer to truncate the folder or file on startup. The default is `processed`.

UNPROCESS

The name of the mail folder or the full path name of the mail file to put unprocessed notification messages. A '-' preceding the name causes the Notification Mailer to truncate the folder or file on startup. The default is `unprocessed`.

TAGFILE

The full path and name of a tag file. The tag file lists strings of text found in unusual messages and the mail status you want to assign to a message response if it contains any of those strings. Unusual messages include bounced or returned messages, auto-reply messages such as those sent by vacation daemons, mass mailing lists, and so on. Since different mail systems vary in how they identify bounced, undeliverable, or otherwise invalid messages, you can use a tag file to specify how your mail system identifies those stray messages and how you want the Notification Mailer to handle those messages should it come across them.

Attention: It is important that you uniquely identify bounced messages from normal responses so that Oracle Workflow does not mistaken a bounced message as an invalid response or a legitimate response as a bounced message.

The format used in the tag file is

Mail_status "Matching string"

where *Mail_status* can be the value: `ERROR`, `INVALID`, `IGNORE`, or `UNAVAIL` and

"Matching string" is the text to look for in the From: line, Subject: line, or body of the message. If

a message is marked `IGNORE`, the Notification Mailer ignores the message and moves it to the processed mail folder/file. If the message is marked with any other mail status, no further processing occurs on the message.

For example, if you want to mark all message responses that contain the string "-- Unsent message follows --" in the subject or body of the message as an error, you can include the following line in your tag file:

```
ERROR "-- Unsent message follows --"
```

Oracle Workflow provides an example tag file called `wfmail.tag`. For the standalone version of Oracle Workflow, the file resides in your Oracle Workflow server directory structure in the subdirectory `res`. For the version of Oracle Workflow embedded in Oracle Applications, the file resides on your server in the `resource` subdirectory under `$FND_TOP`.

► To Run a Perpetual Shell Script for the Notification Mailer

1. If you are running the standalone version of Oracle Workflow, you need to set up a perpetual shell script that restarts the Notification Mailer if it shuts down due to failure. Oracle Workflow provides a sample shell script to restart the UNIX Sendmail or Oracle Office/InterOffice Notification Mailer. The script is called `wfmail.csh` and it is located in the Oracle Workflow `bin` subdirectory on your server.

Note: Use a similar technique to restart the Window NT Notification Mailer.

2. Enter the following command at your operating script prompt to run the shell script:

```
wfmail.csh -f <config_file>
```

Replace `<config_file>` with the full path name of the configuration file that contains the parameters you want to run with the Notification Mailer. The shell script passes all command line arguments directly to the Notification Mailer executable.

Response Processing

You must create three folders or files in your response mail account before starting the Notification Mailer to process responses. The three folders or files serve to hold discarded, unprocessed, and processed messages.

The Notification Mailer does the following to check for response messages:

- Logs into the response mail account.
- Checks for messages. If a message exists, it reads the message, checking for the notification ID and node identifier.
- If the message is not a notification, it moves it to the discard folder.
- If the message is a notification for the current node, it moves the message to the unprocessed folder.
- If the message is a notification, but for the wrong node, it does not move the message so that the Notification Mailer for the correct node can read it later.

The Notification Mailer then opens the unprocessed folder to process each response. For each message, it:

- Retrieves the notification ID.
- Checks to see if the message bounced by referring to a specified tag file, if any. If the message bounced, it reroutes it or updates the notification's status and stops any further processing depending on the specifications of the tag file.
- Checks the Oracle Workflow database for this notification.
 - If the notification does not exist, it moves it to the discard folder.
 - If the notification exists, but is closed or canceled, it moves it to the discard folder.
 - If the notification exists and is open, it verifies the response values with the definition of the message's response attributes in the database. If a response is invalid, it sends an Workflow Invalid Mail message to the recipient role. If the responses are valid, it calls a Respond function to complete the notification response and saves the change to the database.

- Moves the message for the completed notification to the processed folder and closes the unprocessed folder.

The Notification Mailer then truncates the discard and processed folders, if a '-' precedes the discard and process parameters specified in the configuration file, and logs out of the mail and database accounts.

Modifying Your Message Templates

Use the System: Mailer item type in Oracle Workflow Builder to configure the templates that Oracle Workflow uses to send E-mail notifications. The System: Mailer item type has attributes that represent every part of the notification message. You can reorganize the layout of these attributes in each template to customize the E-mail messages sent by the Notification system.

The messages of the System: Mailer item type are not true messages; rather they act as templates for any E-mail messages the Notification system sends. System: Mailer messages determine the basic format of an E-mail notification, including what header information to include, or whether and where to include details such as the message due date and priority.



Warning: Do not add new attributes or delete existing attributes from the messages templates in the System: Mailer item type.

Workflow Open Mail Message

The Notification system uses the Workflow Open Mail message as a template for all E-mail notifications that require a response. The template includes generic instructions on how to respond to a notification. It also includes the following information about a message: message priority, date that a response is due, or if the notification is forwarded from another user, and any comments from the sender or forwarder of the message.

The Workflow Open Mail message has the following message attributes. The values are drawn from the message definition associated with a notification activity.

START_DATE	The date the message is sent.
TO	The role the notification is sent to; the performer.
SUBJECT	The subject line defined in the message.
BODY	The text of the body defined in the message.
COMMENT	Comments added by the sender or the forwarder.
PRIORITY	The priority of the notification message.
DUE_DATE	The date by which a response is required, specified in the notification activity.

- NOTIFICATION** Required notification code used to identify the information in the notification.
- RESPONSE** The user response section as defined by the Respond message attributes in the actual notification message definition.

You can customize the boilerplate text that appears in the body of the Workflow Open Mail template. The body of the Workflow Open Mail template contains the following default text, where attributes preceded by an ampersand (&) are token substituted with runtime values when the notification is sent:

```
Oracle Workflow Notification
&COMMENT
```

```
Response Instructions for &NOTIFICATION
```

```
To submit your response, reply to this message, including
this note with your reply. The first lines of your reply
must be your responses to the notification questions.
Instructions below detail exactly what should be placed on
each line of your reply.
```

```
&RESPONSE
```

```
Notification Details:
&BODY
```

```
Due Date: &DUE_DATE
```

See Also

To Create a Message: page 3 – 31

Workflow Open FYI Mail Message

The Notification system uses the Workflow Open FYI Mail message as a template for all E-mail notifications that do not require a response. The template indicates that the notification is for your information (FYI) and does not require a response. In addition to the message, the

template also includes any comments from the sender or forwarder of the message.

The Workflow Open FYI Mail message has the following message attributes. The values are drawn from the message definition associated with a notification activity.

START_DATE	The date the message is sent.
TO	The role the notification is sent to; the performer.
SUBJECT	The subject line defined in the message.
BODY	The text of the body defined in the message.
COMMENT	Comments added by the sender or the forwarder.
PRIORITY	The priority of the notification message.
DUE_DATE	The date by which a response is required, specified in the notification activity.
NOTIFICATION	Required notification code used to identify the information in the notification.

You can customize the text that appears in the body of the Workflow Open Mail template. The body of the Workflow Open Mail template initially contains the following default text, where attributes preceded by an ampersand (&) are token substituted with runtime values when the notification is sent:

```
Oracle Workflow Notification (FYI)
&COMMENT
```

```
-----
&BODY
```

Workflow Canceled Mail Message

The Workflow Canceled Mail message informs the recipient that a previously sent notification is cancelled. It has the following message attributes, with values that are drawn from the message definition associated with the cancelled notification activity:

START_DATE	The date the original message was sent.
TO	The role the notification is sent to; the performer.
SUBJECT	The subject line of the original message.
BODY	The text of the original message.

COMMENT	Comments added by the sender or the forwarder.
PRIORITY	The priority of the notification message.
DUE_DATE	The date by which a response is required, specified in the notification activity.
NOTIFICATION	Required notification code used to identify the information in the notification.

The body of the Workflow Canceled Mail template initially contains the following customizable text:

You earlier received the notification shown below. That notification is now canceled, and no longer requires your response. You may simply delete it along with this message.

&BODY

Workflow Invalid Mail Message

The Workflow Invalid Mail message gets sent to a user when a user responds incorrectly to a notification. The message describes how to respond to the notification correctly. The message attributes are as follows:

START_DATE	The date the original message was sent.
TO	The role the notification is sent to; the performer.
SUBJECT	The subject line of the original message.
BODY	The text of the original message.
COMMENT	Comments added by the sender or the forwarder.
PRIORITY	The priority of the notification message.
DUE_DATE	The date by which a response is required, specified by the notification activity.
NOTIFICATION	Required notification code used to identify the information in the notification.
RESPONSE	The user response section as defined by the Respond message attributes in the original message definition.

MAIL_ERROR_MESSAGE An error message that the mail program generates if an error occurs upon processing the response.

MAIL_ERROR_STACK An error stack of arguments that the mail program generates if an error occurs upon processing the response. You can provide this information to your support representative if the problem cannot be resolved with a corrected response.

The body of the Workflow Invalid Mail template initially contains the following customizable text:

Oracle Workflow Notification
&COMMENT

NOTE: Your previous response to this message was invalid (see error message below). Please resubmit your response.

Error Message: &MAIL_ERROR_MESSAGE
Error Stack: &MAIL_ERROR_STACK

Response Instructions for &NOTIFICATION

To submit your response, reply to this message, including this original with your reply. This note contains a special NID string that is required to process the response. The first lines of your reply must be your responses to the notification questions. You should enter one line for each response required by the notification, any additional lines will be ignored. You may leave a line blank to accept the default value for that specific response. You must supply a value or a blank line for each question asked. Instructions below detail exactly what should be placed on each line of your reply.

&RESPONSE

Notification Details:

&BODY

Due Date: &DUE_DATE

Workflow Closed Mail Message

The Workflow Closed Mail message informs the recipient that a previously sent notification is now closed. It has the following message attributes, with values that are drawn from the message definition associated with the closed notification activity:

START_DATE	The date the original message was sent.
TO	The role the notification is sent to; the performer.
SUBJECT	The subject line of the original message.
BODY	The text of the original message.
COMMENT	Comments added by the sender or the forwarder.
PRIORITY	The priority of the notification message.
DUE_DATE	The date by which a response is required, specified in the notification activity.
NOTIFICATION	Required notification code used to identify the information in the notification.

The body of the Workflow Closed Mail template initially contains the following customizable text:

```
You earlier received the notification shown below. That notification is now closed, and no longer requires your response. You may simply delete it along with this message.
```

```
-----  
&BODY
```

Workflow Summary Mail Message

The Notification system uses the Workflow Summary Mail message as a template to send a summary of workflow notifications to users and roles that have their notification preference set to 'SUMMARY' in the Oracle Workflow directory service. The Workflow Summary Mail message summarizes all currently open notifications for a given user/role. It has the following message attributes, with values that are drawn from the message definition associated with the open notification activity:

BODY	The subject line of the notification.
-------------	---------------------------------------

USER_NAME The user/role the notification summary is sent to;
the performer.

SYSDATE The current date.

The body of the Workflow Summary Mail template initially contains the following customizable text:

NOTE: Please use a Web browser or Notification form to view notification details.

Summary of Notifications for '&USER_NAME'

&SUMMARY

Workflow Warning Mail Message

The Notification system uses the Workflow Warning Mail message as a template to send a message to user if it receives unsolicited mail from that user. The Workflow Warning Mail message has the message attribute SYSDATE, which identifies the current date.

The body of the Workflow Warning Mail template initially contains the following customizable text, where the last portion of the text includes the mail originally received from the user:

Messages sent to this account are processed automatically by the Oracle Workflow Notification Mailer. The message you sent did not appear to be in response to a notification. If you are responding to a notification, please use the response template that was included with your notification. Take care to include the 'NID' line of the template in your reply. If you are not responding to a notification, please do not send mail to this account.

From: &UFROM
Subject: &USUBJECT

&UBODY

Setting Up Background Workflow Engines

When the Workflow Engine initiates and performs a process, it completes all necessary activities before continuing to the next eligible activity. In some cases, an activity can require a large amount of processing resource or time to complete. Oracle Workflow lets you manage the load on the Workflow Engine by setting up supplemental engines to run these costly activities as background tasks. In these cases, the costly activity is *deferred* by the Workflow Engine and run later by a background engine. The main Workflow Engine can then continue to the next available activity, which may occur on some other parallel branch of the process.

A background engine must also be set up to handle timed out notification activities. When the Workflow Engine comes across a notification activity that requires a response, it calls the Notification System to send the notification to the appropriate performer, then sets the notification activity to a status of 'NOTIFIED' until the performer completes the notification activity. Meanwhile, a background engine set up to handle timed out activities periodically checks for 'NOTIFIED' activities and whether these activities have time out values specified. If a 'NOTIFIED' activity does have a time out value, and the current date and time exceeds that time out value, the background engine marks that activity as timed out and calls the Workflow Engine. The Workflow Engine then resumes by trying to execute a <Timeout> transition activity.

You can define and start up as many background engines as you like to check for deferred and timed out activities. Background engines can be restricted to handle activities associated with specific item types, and within specific cost ranges. A background engine runs until it completes all eligible activities at the time it was initiated. Generally, you should set the background engine up to run periodically by either using a script to restart the background engine periodically (for the standalone version of Oracle Workflow), or scheduling the Background Process concurrent program to resubmit periodically (for the version of Oracle Workflow embedded in Oracle Applications).

See Also

Activity Cost: page 3 – 43

Timeout Transitions: page 4 – 3

To Start a Background Engine

If you are using the standalone version of Oracle Workflow, then use the `WF_ENGINE.BACKGROUND()` API to start up a background engine. Sample scripts that repeatedly run the background engine are provided with the standalone version of Oracle Workflow. See: Background: page 7 – 19.

If you are using the version of Oracle Workflow embedded in Oracle Applications, you can start a background engine by submitting the Background Process concurrent program using the Submit Requests form. See: To Schedule Background Engines: page 2 – 44

Note: Make sure you have a least one background engine that can check for timed out activities and one that can process deferred activities. At a minimum, you need to set up one background engine that can handle both timed out and deferred activities.

To Schedule Background Engines

If you are using the version of Oracle Workflow embedded in Oracle Applications, you can submit the background engine procedure as a concurrent program to schedule different background engines to run at different times. Use the Submit Requests window in Oracle Applications to submit the Workflow Background Process.

► **To Run a Workflow Background Process as a Concurrent Program:**

1. Navigate to the Submit Requests form.
2. Submit the Workflow Background Process concurrent program as a request. See: Submitting a Request, *Oracle Applications User's Guide*.
3. In the Parameters window, enter values for the following parameters:

Item Type Specify an item type to restrict this engine to activities associated with that item type. If you do not specify an item type, the engine processes any deferred activity regardless of its item type.

Minimum Threshold Specify the minimum cost that an activity must have for this background engine to execute it, in hundredths of a second.

Maximum Threshold	Specify the maximum cost an activity can have for this background engine to execute it, in hundredths of a second. By using Minimum Threshold and Maximum Threshold you can create multiple background engines to handle very specific types of activities. The default values for these arguments are 0 and 100 so that the background engine runs activities regardless of cost.
Process Deferred	Specify whether this background engine checks for deferred activities. Setting this parameter to 'Yes' allows the engine to check for deferred activities.
Process Timeout	Specify whether this background engine checks for activities that have timed out. Setting this parameter to 'Yes' allows the engine to check for timed out activities.

Note: Make sure you have a least one background engine that can check for timed out activities and one that can process deferred activities. At a minimum, you need to set up one background engine that can handle both timed out and deferred activities.

4. Choose OK to close the Parameters window.
5. When you finish modifying the run options to define the schedule for the background engine, choose Submit to submit the request.

See Also

Overview of Concurrent Programs and Requests, *Oracle Applications System Administrator's Guide*

Using Standard Request Submission, *Oracle Applications User's Guide*.

To Set Engine Thresholds

To set the thresholds of background engines, specify the minthreshold and maxthreshold arguments when starting the engine. The background engine then only processes activities with costs within your specifications.

The main Workflow Engine threshold is set at 50. Activities with a cost higher than 50 are deferred for background engines to process.

In some cases, you may want to force the engine to defer an activity although the activity's cost is less than fifty. You can do this by altering the Workflow Engine threshold in the PL/SQL stored procedure for a function activity.

The engine threshold is set in an externalized constant called `THRESHOLD`. Include the following line in your PL/SQL procedure to set the WF Engine threshold to a different value:

```
WF_ENGINE.THRESHOLD := n;
```

You should reset the threshold value afterwards in SQL*PLUS or in the next function activity so that other activities are processed as expected.

Adding Custom Icons to Oracle Workflow

Oracle Workflow Builder looks for icons in the Icon subdirectory of the Oracle Workflow area on your PC. The Icon subdirectory is defined in the registry of Oracle Workflow Builder. The Oracle Workflow area is typically the WF20 subdirectory within your ORACLE_HOME for Windows 95 or NT directory structure.

Workflow provides a variety of icons that you can use with your activities and processes. You can add any icon files to this area as long as they are Windows icon files with the *.ico* suffix.

If you want the custom icons that you include in your Oracle Workflow Builder process definition to appear in the Workflow Monitor when you view the process, you must do the following:

- Convert the custom icon files (*.ico*) to *gif* format (*.gif*).
- Copy the *.gif* files to a directory where the Workflow Monitor can access them:
 - For the standalone version of Oracle Workflow —
/*<ORACLE_HOME>*/wf/java/oracle/wf/icons
 - For the Oracle Applications–embedded version of Oracle Workflow — *<java_directory_path>*/oracle/wf/icons

Overview of Oracle Workflow Access Protection

Access protection is a feature that prevents workflow seed data created by a 'seed data provider' from being modified by a 'seed data consumer'. Here, a 'seed data provider' is any organization that creates 'seed data' for other organizations ('seed data consumers') to use in defining and customizing workflow process. In Oracle Workflow, seed data refers to either of the following:

- Workflow object definitions that can and should be customized to meet a certain consumer's needs.
- Workflow object definitions protected against customization because they represent standards that may also be upgraded in the future by the provider.

For example, the Oracle Workflow development team is a provider of seed data called the Standard item type. The Standard item type contains standard activities that can be dropped into any custom workflow process. The development team at your organization's headquarters may create a custom workflow process definition that references activities from the Standard item type. This makes the headquarters team a consumer of the Standard item type seed data.

Now suppose the headquarters team wants to deploy the custom workflow definition that it created to teams at other regional offices. The headquarters team, as seed data providers, may want to do the following:

- Identify certain workflow objects in its custom workflow definition as corporate standards that the regional teams should adhere to and not modify.
- Designate certain objects in its deployed process as customizable for the regional offices to alter to their offices' needs.

The headquarters team can satisfy both requirement using the access protection feature in Oracle Workflow. Access protection lets seed data providers protect certain data as 'read-only', while allowing other data to be customized. Also during a seed data upgrade, access protection lets the seed data provider overwrite any existing protected seed data with new versions of that seed data, while preserving any customizations made to customizable seed data.

Oracle Workflow assigns a protection and customization level to every workflow object definition stored in the database and requires every user of Oracle Workflow to operate at a certain access level. The combination of protection, customization, and access levels make up the access protection feature and determines whether a user can

modify a given workflow object. The level in all three cases, is a numeric value ranging from 0 to 1000 that indicates the relationship between different organizations as providers and consumers of seed data.

The following range of levels are presumed by Oracle Workflow:

0–9	Oracle Workflow
10–19	Oracle Application Object Library
20–99	Oracle Applications development
100–999	Customer organization. You can determine how you want this range to be interpreted. For example, 100 can represent headquarters, while 101 can represent a regional office, and so on.
1000	Public

Access Level

Each user of Oracle Workflow operates the system at a certain access level according to the range of levels listed above. A "user of Oracle Workflow" in this case, represents someone who is operating Oracle Workflow Builder, or the Workflow Definitions Loader program, which loads workflow process definitions from a file into a database. As a seed data provider, you should always operate Oracle Workflow Builder at the same consistent access level because the level you work at affects the protection level of the seed data you create.

You can view your access level as follows:

- In Oracle Workflow Builder, select About Workflow from the Help menu.
- If you are going to run the Workflow Definitions Loader program, check the value for the environment variable `WF_ACCESS_LEVEL` on your workflow server. See: Using the Workflow Definitions Loader: page 2 – 54.

Note: The Workflow Definitions Loader program references the access level stored in the environment variable called `WF_ACCESS_LEVEL`, which you must define when you install Oracle Workflow on your server. If you do not define this environment variable, the Workflow Definitions Loader simply assumes a default access level of 100.

Note: When you install the standalone version of Oracle Workflow on your server, you need to define this variable in an environment file. The default environment file is

APPLSYS.env. If you do not define this environment variable, the Workflow Definitions Loader simply assumes a default access level of 100. Refer to your Oracle Applications Installation Manual for more information about environment files.

Protection Level

Whenever you create a workflow object in Oracle Workflow Builder, you have the option of protecting the object at a certain level. An object's protection level controls whether other users can modify the object based on their access levels.

To change the protection level of an object, display the Access tab of the object's property page. The protection level that you set for an object is dependent on your current access level. You can control access to an object in one of four ways:

- **Allow access to everyone**—By default, all users are allowed access to an object if both 'Preserve Customizations' and 'Lock at this Access Level' are unchecked in the Access tab, that is the protection level is equal to 1000.
- **Limit access to users with access levels equal to your own or higher**—If you check 'Preserve Customizations' in the Options region of the Access tab, you designate the object as being customizable by anyone with an access level equal to or higher than your current access level. You should only mark objects as customizable if you are sure that you will not be providing upgraded versions of this object in the future that would overwrite other user's customizations to it.
- **Limit access to users with access levels equal to your own or lower**—If you check 'Lock at this Access Level', you protect the object and ensure that the object may only be modified by users with an access level equal to or lower than your current access level. Users operating at a higher access level will see a small lock on the workflow object's icon, indicating that the object can be used but not modified. Protect any objects that you want to define as standard components that will not change unless you provide a global upgrade. For this reason, it is important that you always operate at the same consistent access level.
- **Limit access to users with access levels equal to your own**—If you check both 'Lock at this Level' and 'Preserve Customizations' you ensure that the object cannot be modified by anyone other than users operating at your current access level.

Preserve Customizations	Lock at this Access Level	Access Level applied to Object
		Object may be updated by any access level.
X		Object may only be updated by users with access levels equal to or higher than your current access level.
	X	Object may only be updated by users with access levels equal to or lower than your current access level.
X	X	Object cannot be updated by any access level except for your current access level.

Table 2 - 1 (Page 1 of 1)



Attention: If you have installed the beta version of Microsoft's Internet Explorer on your PC, which automatically installs an early version of a file called *comctl32.dll*, you may not see the lock icons appear on the locked objects in Oracle Workflow Builder. To correct this problem, install the production version of Microsoft's Internet Explorer to replace *comctl32.dll* with the latest copy.

The protection and access levels in Oracle Workflow are present to remind you that certain workflow objects should not be modified or should only be modified by someone accessing the tool at an authorized access level. It is not intended as a means of securing or source controlling your workflow objects.



Attention: Most workflow objects provided by Oracle Workflow have a protection level of 0, which means the objects can only be modified by the Oracle Workflow team, operating at an access level of 0. If you attempt to alter your access level to 0 and modify the data anyway, your customizations will not be supported, especially if Oracle Workflow provides an upgrade to the seed data that may overwrite the modifications you make to the originally protected data.

Customization Level

Every workflow object, in addition to having a protection level, also records a customization level equal to your access level when you modify the object and save it to a database or file. For example, if a workflow object is customizable (protection level is 1000), and you customize it at an access level of 100, you now mark the object as having a customization level of 100. The customization level indicates that the object can only be further modified by someone operating at an access level equal to or higher than the customization level. So in this example, you can only customize the object further if your access level is 100 or higher. If you are operating at an access level lower than an object's customization level, you will see a small lock on that workflow object's icon, indicating that the object can be used but not modified

This ensures that a customizable object that has been customized never gets overwritten during a seed data upgrade because the upgrade always occurs with the Workflow Definitions Loader operating at an access level below the customized object's customization level.

Setting Up a Default Access Level

When you install Oracle Workflow Builder on a Microsoft Windows 95 or Windows NT PC, Oracle Installer assigns a default access level that is global to the PC and the operating system you are installing on. Oracle Installer references a file called *level.vrf* to define the default access level for the installation. You can also assign an access level to each individual user on the PC, which overrides the default access level set for the PC. If a user does not have an access level defined, Oracle Workflow Builder assumes the value of the default access level for the PC. The access levels are stored in the Microsoft Windows registry.

If you are deploying Oracle Workflow Builder and workflow seed data to users in other parts of your organization, and you wish to discourage those users from modifying the seed data that you provide, you can have them operate Oracle Workflow Builder at an access level that is higher than the data's protection level. For example if you, as a seed data provider are operating at an access level of 100 and the seed data you create is protected at a level of 100, then you should set the default access level for your users or seed data consumers to be 101 or higher.

You can set your user's default access level in one of two ways in Oracle Workflow Builder:

- If you are installing Oracle Workflow Builder over a LAN, you can edit the file *level.vrf* located in the Windows95 or NT *wrkflw20* installation subdirectory. Change the access level in that file to a level that is higher than your seed data protection level, then have your users install Oracle Workflow Builder. The new default access level is automatically set.
- If your users are installing Oracle Workflow Builder directly from the installation CD, then after installing Oracle Workflow Builder, have them choose About Oracle Workflow Builder... from the Edit menu. In the About Oracle Workflow Builder window, change the Access Level field to a number higher than your seed data protection level, then choose OK.

For the Workflow Definitions Loader program, you set the default access level that the program operates at for downloading process definitions to a file, by defining an environment variable called `WF_ACCESS_LEVEL` and setting its value using the appropriate operating system command.

Caution: Although you can modify your access level, Oracle Workflow does not support any customizations to seed data originally protected at a level 99 or lower. We **STRONGLY RECOMMEND** that you not change your access level to an unauthorized level for modifying protected data.

Using the Workflow Definitions Loader

Rather than use the File Save or File Open menu options in Oracle Workflow Builder, you can also run a program called Workflow Definitions Loader to save or load process definitions from a database or flat file.

When you upgrade your database, use the Workflow Definitions Loader to preserve and back up your process definitions to a flat file. When the database upgrade is complete, use the Loader program again to upload the definitions back into your database. You can also use the Loader program to upgrade your database with a newer version of a process definition or to transfer process definitions to other databases.

The Workflow Definitions Loader program accepts several parameters that vary depending on the mode that you run the Loader program. Following is a summary of the Loader program's various modes and behavior.

Workflow Definitions Loader Behavior

Mode	Data Transfer	Data Protected at Access Level Less Than Loader Access Level	Customized Data
UPGRADE	File to Database	Preserved	Preserved
UPLOAD	File to Database	Preserved	Overwritten
FORCE	File to Database	Overwritten	Overwritten
DOWNLOAD	Database to File	Not Applicable	Not Applicable

Table 2 - 2 (Page 1 of 1)

► **To run the Workflow Definitions Loader for the standalone version of Oracle Workflow:**

1. The Workflow Definitions Loader program is located on your server in the *bin* subdirectory of the Oracle Workflow directory structure.
2. Run the program from your operating system prompt as follows (replacing *<username/password@database>* with the username, password and SQL*Net connect string or alias to your database):

- To apply a seed data upgrade to a database from an input file, type:

```
wfload <username/password@database> <input_file>
```

By using the default upgrade behavior, the Workflow Definitions Loader assumes the access level of the file's creator (seed data provider) and overwrites any objects protected at a level equal to or above the upgrade file's access level. During an upgrade, the Loader program preserves any customizations made to customizable seed data in the database. *<input_file>* represents the name and full path of the upgrade file you are loading.

- To upload process definitions from an input file to a database, type:

```
wfload -u <username/password@database> <input_file>
```

The upload mode is useful to someone who is developing a workflow process. It allows the developer to save definitions to the database without concern that accidental customizations to existing objects might prevent the upload of some process definition elements. The Workflow Definitions Loader uses the access level specified in the input file. *<input_file>* represents the name and full path of the input file you want to upload from.

- To force an upload of the process definitions from an input file to a database regardless of an object's protection level, type:

```
wfload -f <username/password@database> <input_file>
```

<input_file> represents the name and full path of the input file you want to upload from. When using the force option, you should be certain that the process definition in the file is correct as it overwrites the entire process stored in the database. The force option is useful for fixing data integrity problems in a database with a known, reliable file backup. The force option is also useful for loading .wft files from Oracle Workflow Release 1.0 or 1.0.1, which reflect an older data model.

Note: When using the force option to load a .wft file from Oracle Workflow Release 1.0 or 1.0.1 into a database, you must also complete a manual step once the .wft file is loaded. You must associate the lookup types that you load with an item type. To do this, in the Navigator window of Oracle Workflow Builder, drag the lookup types from the independent Lookup Types branch to a Lookup Types branch associated with an item type.

- To download the process definition of one or more item types from a database to an output file, type:

```
wfload [-d <date>] <username/password@database>
<output_file> <item_type1> <item_type2> ...<item_typeN>
```

<output_file> represents the name and full path of the output file you want to write to, and <item_typeN> represents the internal name of each item type you want to download. You can also replace <item_typeN> with '*' to represent all item types (make sure you enclose the asterisk in single quotes). If you specify the -d option with a date (omitting the square brackets), you can download the process definition that was effective at that date. The date must be supplied in the following format: YYYY/MM/DD HH24:MI:SS.

Your output file should have the extension .wft. When you download a process definition, the Loader program sets the output file's access level to be the value stored in the WF_ACCESS_LEVEL environment variable.

► **To run the Workflow Definitions Loader for the version of Oracle Workflow embedded in Oracle Applications:**

1. Navigate to the Submit Requests form in Oracle Applications to submit the Workflow Definitions Loader concurrent program. When you install and set up Oracle Applications and Oracle Workflow, your system administrator needs to add this concurrent program to a request security group for the responsibility that you want to run this program from. See: Overview of Concurrent Programs and Requests, *Oracle Applications System Administrator's Guide*
2. Submit the Workflow Definitions Loader concurrent program as a request. See: Submitting a Request, *Oracle Applications User's Guide*.
3. In the Parameters window, enter values for the following parameters:

Mode	Specify "Download" to download a process definition from the database to a flat file.
	Specify "Upgrade" to apply a seed data upgrade to a database from an input file. The Workflow Definitions Loader assumes the access level of the file's creator (seed data provider) and overwrites any objects protected at a level equal to or above the upgrade file's access level. The Loader

program preserves any customizations made to customizable seed data in the database.

Specify "Upload" to load a process definition from a flat file into the database. The upload mode is useful to someone who is developing a workflow process. It allows the developer to save definitions to the database without concern that accidental customizations to existing objects might prevent the upload of some process definition elements. The Workflow Definitions Loader uses the access level defined by the input file to upload the process definitions from the file and therefore will overwrite objects in the database that are protected at a level equal to or higher than that file's access level.

Specify "Force" to force an upload of the process definitions from an input file to a database regardless of an object's protection level. You should be certain that the process definition in the file is correct as it overwrites the entire process stored in the database. The Force mode is useful for fixing data integrity problems in a database with a known, reliable file backup and for loading .wft files from Oracle Workflow Release 1.0 or 1.0.1, which reflect an older data model.

Note: When using the Force mode to load a .wft file from Oracle Workflow Release 1.0 or 1.0.1 into a database, you must also complete a manual step once the .wft file is loaded. You must associate the lookup types that you load with an item type. To do this, in the Navigator window of Oracle Workflow Builder, drag the lookup types from the independent Lookup Types branch to a Lookup Types branch associated with an item type.

- | | |
|------------------|---|
| File | Specify the full path and name of the file that you want to download a process definition to, or upgrade or upload a process definition from. |
| Item Type | If you set Mode to "Download", use the List button to choose the item type for the process definition you want to download. |

Note: When you submit the Workflow Definitions Loader from the Submit Requests form to download process definitions to a file, you can only specify to download one item

type at a time. If you wish to download multiple or all item types simultaneously, you should submit the Workflow Definitions Loader concurrent program from the command line. See Step 6 below for details.

4. Choose OK to close the Parameters window.
5. When you finish modifying the print and run options for this request, choose Submit to submit the request.
6. Rather than use the Submit Requests form, you can also run the Workflow Definitions Loader concurrent program from the command line by entering the following commands:

To upgrade— `WFLOAD apps/pwd 0 Y UPGRADE file.wft`

To upload— `WFLOAD apps/pwd 0 Y UPLOAD file.wft`

To force— `WFLOAD apps/pwd 0 Y FORCE file.wft`

To download— `WFLOAD apps/pwd 0 Y DOWNLOAD
file.wft ITEMTYPE1 [ITEMTYPE2 ...
ITEMTYPEN]`

Replace *apps/pwd* with username and password to the APPS schema, replace *file.wft* with the file specification of a workflow process definition file, and replace *ITEMTYPE1*, *ITEMTYPE2*, ... *ITEMTYPEN* with the one or more item type(s) you want to download. You can also download all item types simultaneously by replacing *ITEMTYPE1* with *'*'* (make sure you enclose the asterisk in single quotes).

A file specification is specified as:

```
@<application_short_name>:[<dir>/.../]file.ext
```

or

```
<native path>
```

CHAPTER

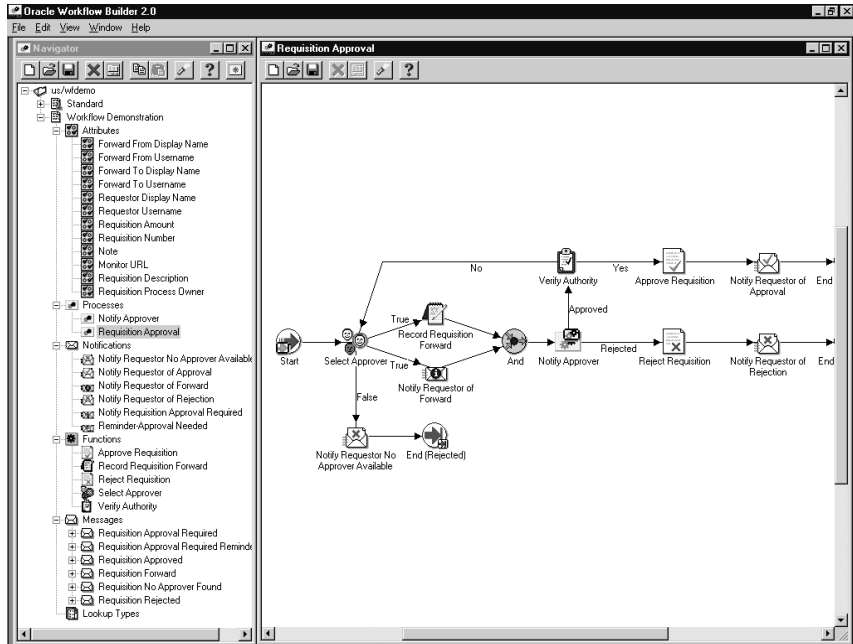
3

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.

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 3 – 6.



Note: When you use Copy from the Edit menu to copy objects from Navigator window to the clipboard, then exit Oracle Workflow Builder and restart it again, you will no longer be able to paste the object from the clipboard back into the Navigator.

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

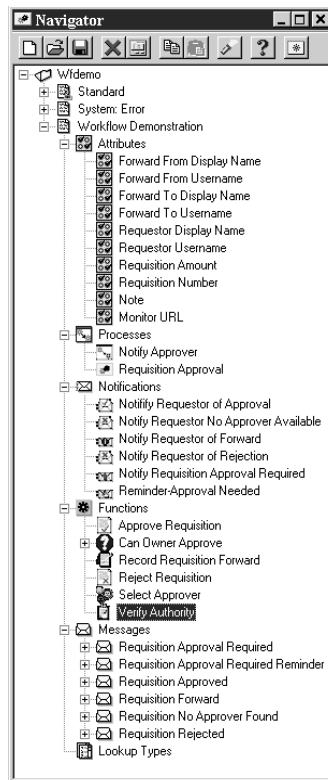
The Navigator window displays a navigator tree hierarchy for each data store that you open or load into Oracle Workflow Builder. A data store (primary branch) is a database connection or flat file that holds your workflow process definition. Within each data store, there is at least one item type heading (secondary branch) that represents the grouping of a particular set of processes and its components. The following six 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 – 14.
- **Processes**—lists the process activities or workflow process definitions for the current item type. See: *Process Window*: page 4 – 2 and *Activities*: page 3 – 41.
- **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 – 41.
- **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 – 41.
- **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 – 30.
- **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 – 26.

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 0 – 6.



► **To Find an Object in the Navigator Tree**

The image shows a 'Search' dialog box with the following fields and options:

- Search Text:** A text input field containing 'Requisition Amount'.
- Field:** A section with two checkboxes: Display Name and Internal Name.
- Object Type:** A section with seven checkboxes: Item Type, Message, Function, Lookup Type, Notification, Lookup Code, and Attribute.
- All Objects:** A checkbox at the bottom of the Object Type section, currently unchecked.
- Buttons:** 'Search' and 'Close' buttons at the bottom of the dialog.

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.
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 Display Name or Internal Name field of the object's property page.
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 happen, 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. For an example of a predefined process, see: Requisition Approval: page 10 – 2.

Versioning and Dates of Effectivity

When you create a process definition, Oracle Workflow Builder assigns a new version number to an activity if you make changes to it. 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 point in the future.

Note that Oracle Workflow Builder does not maintain version information for objects 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.

Using the Edit Button in a Property Page

When you create an object in Oracle Workflow Builder, you must enter information in the object's property page to define the object. 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 other property pages in Oracle Workflow Builder, an Edit button will appear to the right of that poplist. When you select a value in the poplist, you can choose its 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 value you enter can be <None> or any lookup type that is loaded in your current data store. These lookup types are presented to you in a poplist in the Result Type field. If you select a lookup type, you can then choose the Edit button next to the Result Type field 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 or modify a process definition:**

1. To start Oracle Workflow Builder, double-click on the Oracle Workflow Builder icon in the Oracle for Windows NT or Oracle for Windows 95 program group. If you are using Windows 95 or NT 4.0, you can also select the Oracle Workflow Builder icon from the appropriate program folder from the Start menu.
2. Choose New from the File menu to create a workspace for your new process definition or open a connection to the database or file that contains the process definition you want to modify: page 3 – 9.
3. Create a new item type: page 3 – 17 or select and expand the item type associated with the process definition you want to modify.
Create an item type that classifies the work item to be managed by the process. You can also define item type attributes to fully describe your item and have the activities in your process refer to these attributes for information about the item. See: To Define an Item Type or Activity Attribute: page 3 – 19.
4. Create new lookup types: page 3 – 27.
Before you define an activity, define the lookup type that represents your activity's Result Type. 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, activity, or message attributes.
5. Create new messages: page 3 – 31.
Before you define a notification activity, 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. See: To Define a Message Attribute: page 3 – 34.

6. Create a new process activity, notification activity or function activity.

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. See: Activities: page 3 – 41 and To Define an Item Type or Activity Attribute: page 3 – 19.

7. Diagram the process: page 4 – 4.

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.

8. In a database accessible by your Oracle Workflow server, create the PL/SQL stored procedures called by your function activities. You can do this through SQL*PLUS or the Oracle Procedure Builder. See: Workflow APIs: page 7 – 3.

Opening and Saving Item Types

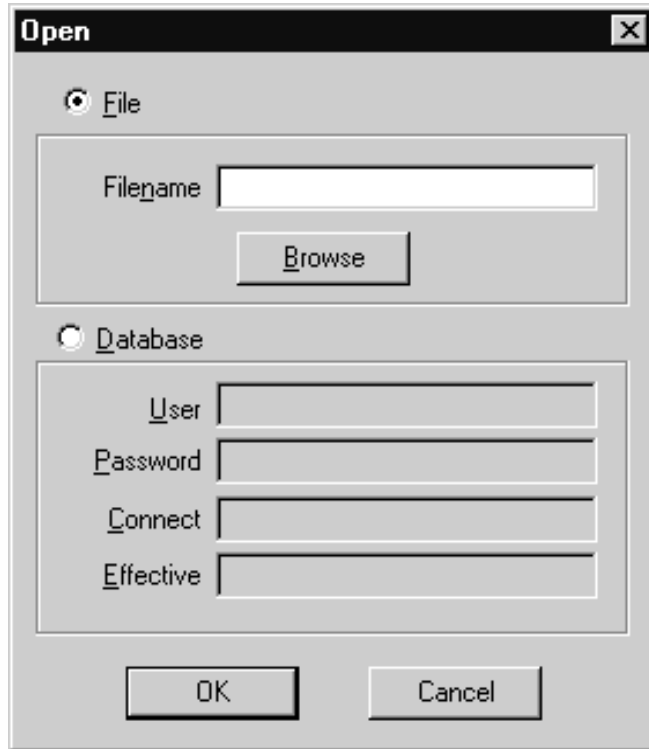
All processes are associated with an item type. An item type can include one or more processes. You can save an item type to a database or to a flat file. When you save your work to a database, you actually save everything in the current data store that has been modified. When you save your work to a 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.



Attention: 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.

► **To Access Process Definitions in an Existing Data Store:**

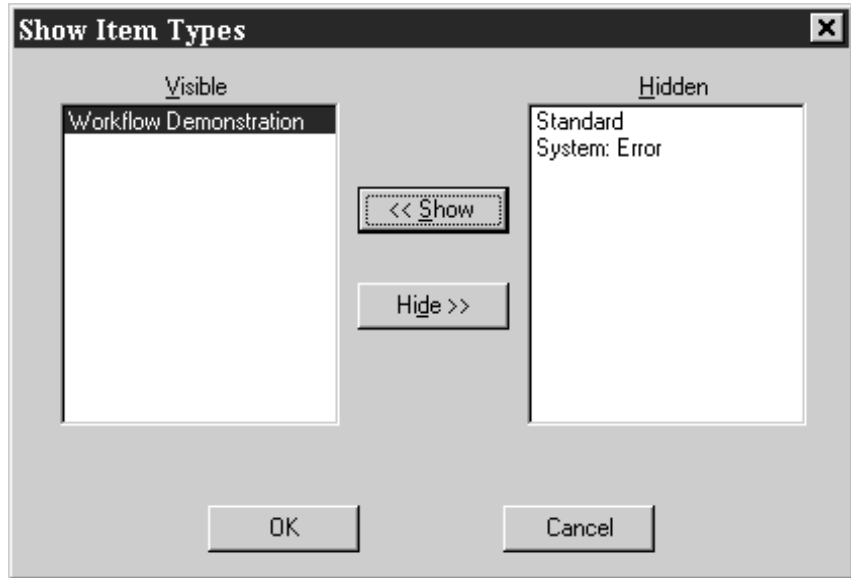
1. To start Oracle Workflow Builder, double-click on the Oracle Workflow Builder icon in the Oracle for Windows NT or Oracle for Windows 95 program group. If you are using Windows 95 or NT 4.0, you can also select the Oracle Workflow Builder icon from the appropriate program folder from the Start menu. In Oracle Workflow Builder, select Open... from the File menu.



2. Select database or file to connect to the source containing the item type that your process definition is associated with.
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 95 or Windows NT 4.0 Explorer or Microsoft Windows NT File Manager into the navigator tree to open that file in Oracle Workflow Builder.

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.
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 use to specify the date and time depends on the date and time preferences defined in the Regional Settings of your Windows Control Panel.



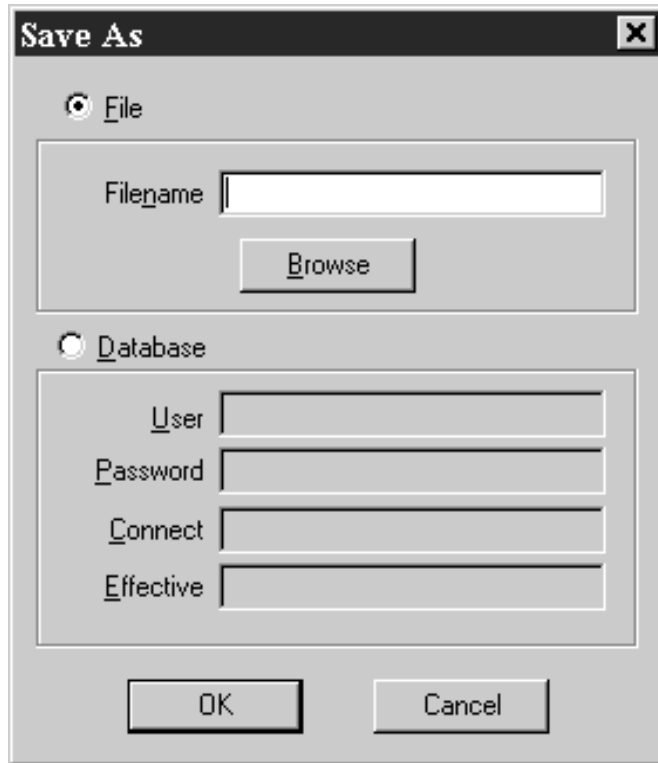
6. If multiple item types exist in the data store, the Show Item Types window appears. Select from the Hidden list, the item type(s) you want to view, and choose <<Show to move it into the Visible list. Choose OK to load these item types into the navigator tree.
7. If at any time you want to view and 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.
8. When you finish working, choose Save from the File menu to preserve your changes and make them effective immediately.

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.

Note: When you copy item types to the new store, you may get validation errors due to foreign key references. You should 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.



Attention: 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 and 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. See: To Set the Access Level for an Object: page 3 – 25 and Using the Workflow Definitions Loader: page 2 – 54.



9. 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: page 7 – 6.

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.

10. Use Close Store from the File menu to close your connection to the current database or file data store. You are prompted to save your work.

11. Choose Exit from the File menu to exit Oracle Workflow Builder.



Attention: 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:

```
wfbldr20 <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:

```
wfbldr20 -c <username/password@connect>
```

Note: If you run Oracle Workflow Builder in Microsoft Windows 95 or Windows NT 4.0, 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.

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:

```
wfbldr20 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:

See Also

Using the Workflow Definitions Loader: page 2 – 54

Creating a Shortcut to a Workflow Process: page 4 – 14

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 times it makes sense to define an item type so that it describes the item being managed by your workflow process. For example, a 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 – 17.

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 referred 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 properly manage the item. 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 .

Attribute Types

There are nine types of attributes, 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.
Number	The attribute 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	<p>The attribute value is the Oracle Applications internal function name of a form and any optional form parameters. This attribute type is not relevant for the standalone version of Oracle Workflow. It is useful only if you have the Oracle Applications Notifications Viewer form.</p> <p>The value must be entered using the following format:</p> <p><i>function_name:parameter1=value1 parameter2=value2 ... parameterN=valueN</i></p> <p><i>valueN</i> can be a text string, enclosed in quotes (" ") or can be token substituted with another predefined item type attribute in any of the following ways:</p> <ul style="list-style-type: none">• <i>parameterN="&item_type_attribute"</i>• <i>parameterN="Value &item_type_attribute"</i> where <i>&item_type_attribute</i> represents the rest of the value. <p>A form attribute is passed to the Notification Viewer window to let a user drill down to the form to complete an activity or to see additional information related to the activity. See: Overview of Menus and Function Security, <i>Oracle Applications Developer's Guide</i>.</p>
URL	The attribute value is a Universal Resource Locator (URL) to a network location that a user can access when viewing notifications from the Notifications Web page. The user can complete an activity or see

additional information related to the activity by accessing that URL.

Document

The attribute value is an attached document. You specify the name of the document management system and a document reference. In the future, you can specify any of the following document types in the default value field:

- PL/SQL—a document representing data from the database, generated from a PL/SQL procedure. Specify as `plsql:<procedure>/<document_identifier>`. The PL/SQL procedure must follow a standard API format. See: Standard API for a "PL/SQL" Document: page 6 – 9.
- Http—a standard URL. Specify as `http://<host>/<document_path>`.
- File—a file accessible from the Oracle Workflow server filesystem. Specify as `file:<file_path>`.
- External document management system—a document managed by an external document management system. Specify as `<doc_management_system>:<document_reference>`.



Attention: Only the PL/SQL document type is currently supported.

Role

The attribute value is a role name. You must initially load the roles from the database to display a list of roles to choose from. See: Roles: page 4 – 16.

Attribute

The attribute value is the name of another existing item type attribute that you want to maintain references to in a process.

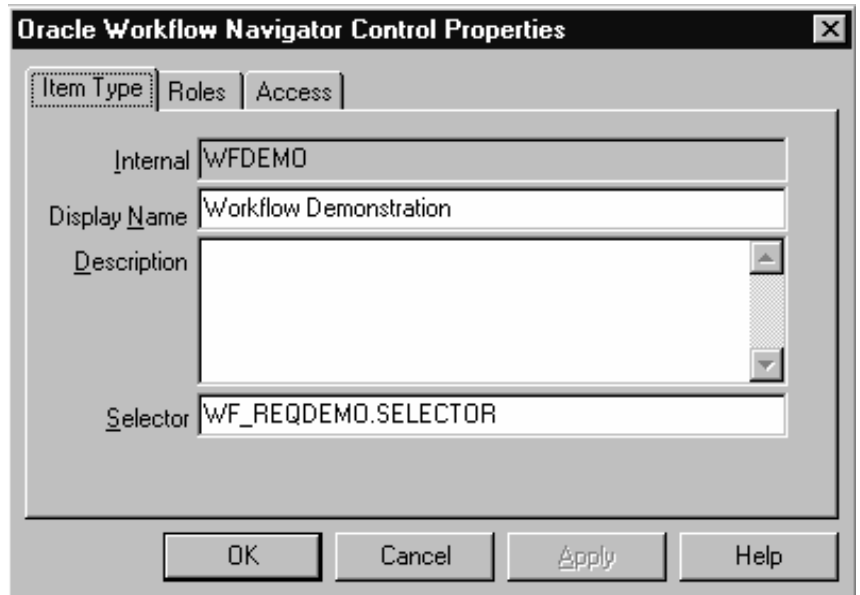
Item Type Selector Function

If your item type has or will have more than one process activity associated with it, define a PL/SQL function that will determine 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 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 during the execution of a process. See: Standard API for an Item Type Selector or Callback Function: page 6 – 5.

► **To Create an Item Type**



1. If you do not already have a data store open, 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.
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.



Attention: You cannot update the internal name for an item type once it is defined.

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

3. The translatable Display Name should be longer and more descriptive. You can also supply a description of the item type.
4. 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 – 5.
5. Choose Apply to save your changes.
6. 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.)
7. Select the Access tab page to set the access levels allowed to modify this item type. See: Allowing Access to an Object: page 3 – 24.
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. 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.
10. 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 your function and notification activities. See: To Define an Item Type or Activity Attribute: page 3 – 19.

See Also

Using the Edit Button in a Property Page: page 3 – 6

► To Define an Item Type or Activity Attribute

The screenshot shows the 'Oracle Workflow Navigator Control Properties' dialog box. It has two tabs: 'Attribute' and 'Access'. The 'Attribute' tab is selected. The fields are as follows:

- Item Type: Workflow Demonstration
- Internal Name: REQUISITION_AMOUNT
- Display Name: Requisition Amount
- Description: Requisition amount
- Type: Number
- Format: 9,999,999,999.99
- Default Value: Constant

Buttons at the bottom: OK, Cancel, Apply, Help.

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

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

An Attribute property page appears in both cases.

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



Attention: You cannot update the internal name for an attribute once it is defined.

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

3. Enter the 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.
6. Depending on the Type of your attribute, provide the following information:

Text	Specify the maximum length of the text attribute.
Number	Optionally provide a format mask for your number.
Date	Optionally supply a format mask for the date.
Lookup	Choose the name of a predefined Lookup Type from which to draw values.
URL	Specify a Universal Resource Locator (URL) to a network location in the Default Value field.
Form	<p>This attribute is relevant only with the version of Oracle Workflow embedded in Oracle Applications.</p> <p>Specify the developer function name of a form and any optional form parameters in the Default Value field. See: Overview of Menus and Function Security, <i>Oracle Applications Developer's Guide</i>. The default value must be entered using the following format:</p> <p><i>function_name;parameter1=value1 parameter2=value2 ... parameterN=valueN</i></p> <p><i>valueN</i> can be a text string, enclosed in quotes (" ") or can be token substituted with the value from another predefined item type or message attribute in any of the following ways:</p> <ul style="list-style-type: none"> • <i>parameterN="&item_type_attribute"</i> • <i>parameterN="Value &item_type_attribute"</i> where <i>&item_type_attribute</i> represents the rest of the value.
Document	<p>Enter the name of the document management system and a document reference. In the future, you can specify any one of the following document types in the default value field:</p> <ul style="list-style-type: none"> • PL/SQL—a document representing data from the database, generated from a PL/SQL procedure. Specify as <i><procedure>/<document_identifier></i>. The PL/SQL procedure must follow a standard API format. See: Standard API for a "PL/SQL" Document: page 6 – 9.

- **Http**—a standard URL. Specify as `http://<host>/<document_path>`.
- **File**—a file accessible from the Oracle Workflow server filesystem. Specify as `<file_path>`.
- **External document management system**—a document managed by an external document management system. Specify as `<doc_management_system>:<document_reference>`.



Attention: Currently, only the PL/SQL document type is supported.

Role	Specify a role name. You must initially load the roles from the database to display a list of roles to choose from. See: Roles: page 4 – 16.
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.

7. For item type attributes, you may specify a default value that is a constant.

For activity attributes, the default value may be a constant or an item type attribute. If the default references a value 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 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, but 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 – 24.
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 function 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.



Attention: The order that 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.

See Also

Using the Edit Button in a Property Page: page 3 – 6

► 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.



Attention: 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.

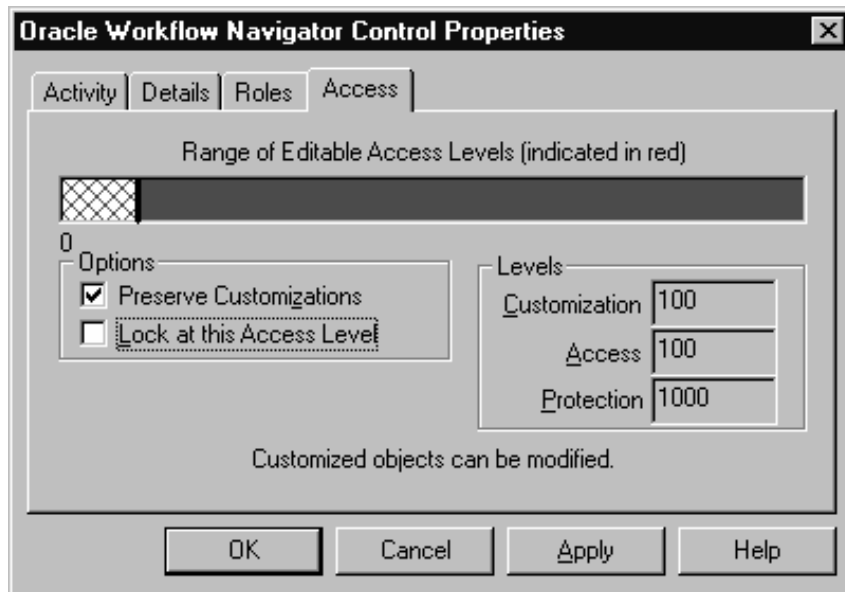
When you are done, choose OK.

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

See Also

Using the Edit Button in a Property Page: page 3 – 6

Allowing Access to an Object



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: page 2 – 48.

The indicator bar can be shaded solid red, or shaded with any combination of solid red and crosshatch red. If the "Allow modifications of customized objects" checkbox 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 red and crosshatch red areas. The levels depicted by red 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 red 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 – 25 and Using the Workflow Definitions Loader: page 2 – 54.

► **To Set the Access Level for an Object**

1. 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.

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 red crosshatches in the indicator bar.

Table 3 – 1 (Page 1 of 2)

Selected Checkbox	Resulting Levels	Levels Allowed to Modify the Object
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 red crosshatches in the indicator bar.

Table 3 – 1 (Page 2 of 2)

- 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 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.

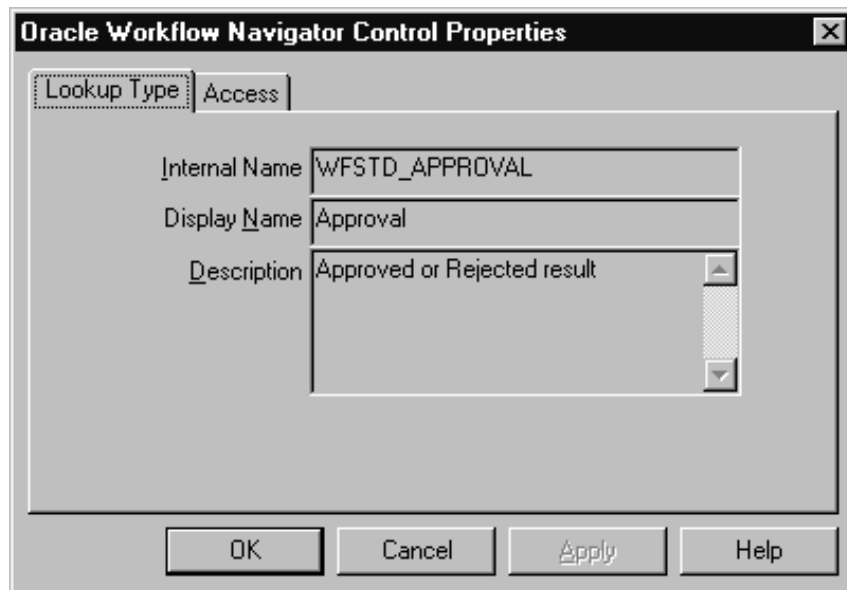
Lookup Types

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, 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. See: To Create Lookup Types: page 3 – 27.

► **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.
2. Lookup types have an all-uppercase Internal Name with no 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.



Attention: You cannot update the internal name for a lookup type once it is defined.

Caution: Do not include colons ":" or 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 – 24.

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 – 28.

► **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.
2. Enter an Internal Name with no 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.



Attention: You cannot update the internal name for a lookup code once it is defined.

Caution: Do not include colons ":" or 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 step 1 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 the lookup type back to the same item type or to a different data store. Its property page appears for you to enter a unique internal and display name for the lookup type. When you are done, choose OK.

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

Note: If you drag and drop a lookup type back to the same item type or to a different data store, the behavior in Step 2 occurs. If you drag and drop a lookup type to another data store, you actually move it to the new data store.

► **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. See: *To Create a Message*: page 3 – 31.

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 as tokens that are replaced with runtime values when the message is delivered.

When you define a message, you can include message attributes that reference item type attributes in the subject and body text. In addition, you can create message attributes that generate a response section unique to the message. As an example, you can create a message attribute called RESULT that prompts a recipient for a response value. The Workflow Engine then uses that response value to determine what the next eligible activity in the process is. See: *To Define a Message Attribute*: page 3 – 34.

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 activity to use that message.

Send and Respond Message Attributes

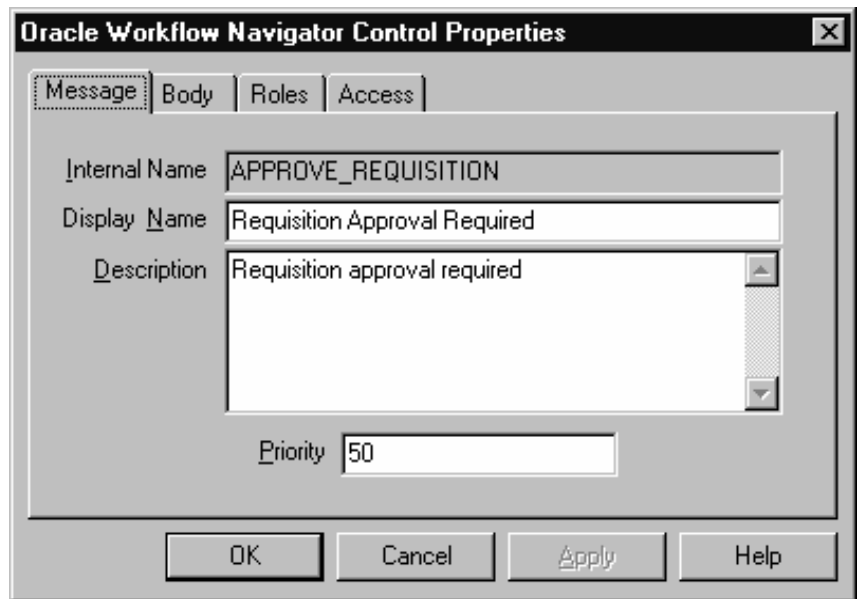
Once you create a message, you can define as many message attributes as necessary for that message. Message attributes are listed beneath a message in the navigator tree.

The source (Send or Respond) of a message attribute determines how the message attribute is used. Message attributes defined with a source of 'Send' are generally included in the message's subject and/or body to provide information when the message is sent. Specify the data type of the message attribute. 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 include a message attribute in a message's subject or body so that it gets token substituted with a runtime value when the message is sent, specify the internal name of the message attribute using the format &MESGATTR within the subject or body text.

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 and you can 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.

Note that if a notification activity has a Result Type defined, that is, it expects a result that determines what the next eligible activity is, you must define a special 'Respond' message attribute in the message that the notification sends. This special 'Respond' message attribute must have an internal name called RESULT, and must be a lookup type to provide a list of possible response values. The response values should originate from the same lookup type used to define the activity's Result Type. This ensures that the response value returned by RESULT is identical to one of the possible results that the notification activity can branch on.

► **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.

2. Provide an internal name for the message that is all uppercase with no 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.



Attention: You cannot update the internal name of a message once it is defined.

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

3. Enter a value between 1 (high) and 99 (low) for the default priority of the message. The priority is for the recipient's information only. It does not affect the processing or delivery of the message.

Note: You can override this default priority value with a priority value that is dynamically set at runtime. See: To Dynamically Set the Priority of a Notification Activity: page 3 – 39.

4. Choose Apply to save your changes.
5. Select the Roles tab page to specify the roles that have access to this message. (This functionality will be supported in a future release.)
6. Select the Access tab page to set the access levels allowed to modify this message. See: Allowing Access to an Object: page 3 – 24.

The screenshot shows the 'Oracle Workflow Navigator Control Properties' dialog box with the 'Body' tab selected. The 'Subject' field contains the text 'Requisition &REQUISITION_NUMBER for &REQUISITION_'. The 'Body' field contains the text 'The following requisition requires your approval. Requisition Number: &REQUISITION_NUMBER Requisition Amount: &REQUISITION_AMOUNT Forwarded From: &FORWARD_FROM_DISPLAY_NAME'. The dialog box has buttons for 'OK', 'Cancel', 'Apply', and 'Help' at the bottom.

7. Select the Body tab to display the Body property page of the message.
8. Enter a description of the message in the Subject field. The subject can include message attributes that get replaced with runtime values when the message is delivered. To include a message attribute, use an "&" followed by the message attribute's internal name. See: Send and Respond Message Attributes: page 3 – 30 and To Define a Message Attribute: page 3 – 34.



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

9. Enter the message body in the Body field. This is the text of the message delivered to the recipient.

You can include message attributes that get replaced with runtime values when the message is delivered. Use an "&" followed by the message attribute's internal name.
10. 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.
11. 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.
12. You must now define all the message attributes that you have included in the subject and body of this message.
13. 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 set. In the Default Value region, select Item Attribute, then use the poplist icon to select the name of the item type attribute that this message attribute references.
14. You can also create message attributes that are not based on existing item type attributes. See: To Define a Message Attribute: page 3 – 34.



Attention: If you want your message to prompt the recipient for a response and that response determines what the next eligible activity in the process is, then you must define a special message attribute that has an internal name called **RESULT**.

RESULT should have a source of 'Respond' and should be of type Lookup, where its Lookup Type is the same as the notification activity's Result Type. This ensures that a response to the notification is identified as the notification activity's result.

► **To Define a Message Attribute**

The screenshot shows the 'Oracle Workflow Navigator Control Properties' dialog box with the 'Attribute' tab selected. The fields are as follows:

Message	Requisition Approval Required
Internal Name	REQUISITION_AMOUNT
Display Name	Requisition Amount
Description	Requisition amount
Type	Number
Source	Send
Format	9,999,999,999.99
Default Value	Item Attribute Requisition Amount

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.

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



Attention: You cannot update the internal name for an attribute once it is defined.

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

3. Enter a Display Name. This is the name that appears in the navigator tree. If this message attribute is going to have a source of 'Respond', then this display name is also used as the prompt for a response value.

4. Enter an optional description. If this message attribute is going to have a source of 'Respond', then use this field to elaborate on response instructions.
5. Select the data type of the attribute.
6. Depending on the Type of your attribute, provide the following information:

Text Specify the maximum length of the text attribute.

Number Optionally provide a format mask for your number.

Date Optionally supply a format mask for the date.

Lookup Choose the name of a predefined Lookup Type from which to draw values.

URL Specify a Universal Resource Locator (URL) to a network location in the Default Value field.

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

Specify the developer function name of a form and any optional form parameters in the Default Value field. See: Overview of Menus and Function Security, *Oracle Applications Developer's Guide*. The default value must be entered using the following format:

*function_name:parameter1=value1 parameter2=value2
... parameterN=valueN*

valueN can be a text string, enclosed in quotes (" ") or can be token substituted with the value from another predefined item type or message attribute in any of the following ways:

- *parameterN= "&item_type_attribute"*
- *parameterN= "Value &item_type_attribute"* where *&item_type_attribute* represents the rest of the value.

Document Enter the name of the document management system and a document reference. In the future, you can specify any one of the following document types in the default value field:

- PL/SQL—a document representing data from the database, generated from a PL/SQL procedure. Specify as `<procedure>/<document_identifier>`. The PL/SQL procedure must follow a standard API format. See: Standard API for a "PL/SQL" Document: page 6 – 9.
- Http—a standard URL. Specify as `http://<host>/<document_path>`.
- File—a file accessible from the Oracle Workflow server filesystem. Specify as `<file_path>`.
- External document management system—a document managed by an external document management system. Specify as `<doc_management_system>:<document_reference>`.



Attention: Currently, only the PL/SQL document type is supported.

Role

Specify a role name. You must initially load the roles from the database to display a list of roles to choose from. See: Roles: page 4 – 16.

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.



Attention: 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.

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

For 'Send' message attributes, the Display Name of the attribute appears in the notification only if the attribute is of type URL to describe what the URL drills down to.

For 'Respond' message attributes, the Display Name and Description of the attribute make up the Response section of the notification message. Use the Description field to describe the response expected, and the Display Name field to specify the response prompt.

'Respond' message attributes of type Date, Number, Text, Document or Role prompt the notification recipient to respond

with a date, number, text value, role (internal or display name), or document, respectively.

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

'Send' and 'Respond' message attributes of type URL appear only when a notification is viewed using the Notification Web page. A 'Respond' message attribute of type URL prompts the notification recipient to link to another HTML page to complete the notification response. The HTML document that the URL links to takes over the entire browser page, and must include a call to the Workflow Engine *CompleteActivity()* API to inform the workflow engine when the notification response is complete.

'Send' and 'Respond' message attributes of type Form appear only when a notification is viewed using the Notification Viewer form in Oracle Applications. The Reference button on the Notification Viewer form is enabled if a notification message includes a 'Send' message attribute of type Form. The notification recipient can choose the Reference button to drill down to the form defined by the message attribute.

If a message includes a 'Respond' message attribute of type Form, the Respond button in the Notification Viewer form drills down directly to the designated form. This form 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: page 7 – 8.

8. For message attributes, the default value may be a constant or an item type attribute. If the default references a value 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, but all other message attribute types must match the item attribute type exactly.

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.

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

10. 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.



Attention: The order that you list 'Respond' message attributes in the navigator tree correlate 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.

11. Following is an example of how the Notification System uses a template to generate the Response section of an E-mail notification using a sample set of 'Respond' message attributes.

Sample 'Respond' Message Attributes

Internal Name	Type	Format/Lookup Type	Display Name	Description
RESULT	lookup	WFSTD_APPROVAL	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.

Table 3 - 2 (Page 1 of 1)

This boilerplate text is used to generate the Response section:

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 one of the following statements, if the message attribute matches one of these data types:

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

Resulting Response Section in an E-mail Notification

<p>Enter the Action on line 1. Do you approve? Value must be one of the following:</p> <p>Approve</p> <p>Reject</p> <p>Enter the Review Comments on line 2. Value must be 2000 bytes or less.</p> <p>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".</p> <p>Enter the Maximum Amount on line 4. This is the maximum approved amount. Value must be a number.</p>

Table 3 – 3 (Page 1 of 1)

See Also

Using the Edit Button in a Property Page: page 3 – 6

To Respond to a Notification by E-mail: page 8 – 7

► To Dynamically Set the Priority of a Notification Activity

1. Create an item type attribute of type number which represents the priority of your notification activity. See: Item Type Attributes: page 3 – 14 and To Define an Item Type or Activity Attribute: page 3 – 19.
2. Create a new activity attribute for the notification activity that you want to dynamically set a priority for. Select the notification

activity in the navigator tree, then choose New Attribute from the Edit menu.

3. In the Attribute property page that appears, enter #PRIORITY in the Internal Name field (all uppercase with no spaces). When the Workflow Engine sees a notification activity attribute called #PRIORITY, it uses that attribute's value as the priority of that notification's message. If such an attribute does not exist or if its value is null, it uses the default priority defined in the message's property page.
4. Enter a Display Name. This is the name that appears in the navigator tree.
5. Enter an optional description.
6. Select number in the Type field. You can leave the format field blank.
7. For the Default Value, choose Item Attribute, then use the poplist field to choose the item type attribute that you defined in step 1.
Since this activity attribute references an item type attribute, its value can be dynamically set at runtime.
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.

See Also

Using the Edit Button in a Property Page: page 3 – 6

► 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.

Activities

An activity is a unit of work that contributes toward the accomplishment of a process. An activity can be a notification, a function, 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 to perform an automated function. A process activity is a modelled workflow process, which can be included as an activity in other processes to represent a sub-process.

Activities are organized beneath their respective 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.

Each activity is depicted as an icon in a process diagram. Ideally, a process diagram should contain different activities that are represented by different icons.

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 – 2.

Oracle Workflow also provides an item type called System:Error that includes activities you can use to create a custom error process. You can assign your custom error process to a process activity, so that if an error occurs, Oracle Workflow knows what to do to handle the error. See: Default Error Process: page 5 – 16.

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 that requires a response, 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 – 44.

You must specify the performer of a notification activity in the Process window when you include the notification as an activity 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 – 7.

When you define a notification activity, you have the option of expanding the role that you send the notification to. 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. However, if one user in that role responds or closes that notification, the notification will be removed from the notification queue of all other users in that role.

If you expand the role for a notification activity, Oracle Workflow sends an individual copy of that notification message to each user in the role and that notification will remain in any user's notification queue until that user responds or closes the notification. Generally, you want to expand the role for a notification activity if the activity sends out a broadcast-type message that you want all users of that role to see. You can also use the Expand Roles feature to create your own custom vote tallying activity. Voting Activity: page 5 – 8.

Function Activity

A function activity is defined by the PL/SQL stored procedure that it calls and is executed directly by the Workflow Engine. Function activities are typically used to perform fully automated steps in the process, and as PL/SQL stored procedures, they accept standard arguments and can return a completion result. See: To Create a Function Activity: page 3 – 46.

If you externalize a parameter for the stored procedure, you can expose that parameter as an activity attribute. The activity attribute's value can be set in the Process window when you define that activity as a node in your process. Note that these activity attributes can be accessed only by the current activity and are not global like item type attributes. See: To Define Activity Attribute Values: page 4 – 9.

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 sub-process. 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 – 48.

Activity Cost

Each function 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 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.



Attention: 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 costs 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 Workflow Engines: page 2 – 43 and To Set Engine Thresholds: page 2 – 45.

► **To Create a Notification Activity**

The screenshot shows the 'Oracle Workflow Navigator Control Properties' dialog box. It has four tabs: 'Activity', 'Details', 'Roles', and 'Access'. The 'Activity' tab is active. The fields are as follows:

- Internal Name: NTF_REQUESTOR_OF_APPROVAL
- Display Name: Notify Requestor of Approval
- Description: Notify requestor that requisition is approved
- Result Type: <None> (dropdown menu)
- Message: Requisition Approved (dropdown menu)
- Icon: NTF_GOOD.ICO (dropdown menu)
- Expand Roles:
- Browse: (button with envelope icon)

At the bottom of the dialog are buttons for OK, Cancel, Apply, and Help.

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.

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 display the Activity property page.

2. A notification activity must have an Internal Name (all uppercase and no 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.



Attention: You cannot update the internal name of an activity once it is defined.

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

3. Indicate the result type (a predefined Lookup Type) of 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 – 27.

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. 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: page 2 – 47.

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. Select the name of the message you want this notification to send. See: To Create a Message: page 3 – 31.
6. 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 – 41.

If you check Expand Roles and you assign a message to this notification activity that includes a message attribute with a source of Respond, you see an additional Function field appear. Use this 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 5 – 8.

7. Choose Apply to save your changes.
8. Select the Details tab to display and modify the Details property page of the activity. See: To Define Optional Activity Details: page 3 – 50.
9. 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.)
10. Select the Access tab page to set the access levels allowed to modify this notification. See: Allowing Access to an Object: page 3 – 24.
11. If you wish to dynamically set the priority of the message that this notification sends (and override the message's default priority), create a special activity attribute whose internal name is

#PRIORITY. See: To Dynamically Set the Priority of a Notification Activity: page 3 – 39.

See Also

Using the Edit Button in a Property Page: page 3 – 6

► To Create a Function Activity

The screenshot shows the 'Oracle Workflow Navigator Control Properties' dialog box. It has four tabs: 'Activity', 'Details', 'Roles', and 'Access'. The 'Activity' tab is active. The fields are as follows:

- Internal Name: SELECTAPPROVER
- Display Name: Select Approver
- Description: Select the appropriate approver from the employee approval
- Function: WF_REQDEMO.SELECTAPPROVER
- Result Type: Boolean (dropdown menu)
- Cost: 0.00
- Icon: FINDUSER.ICO (dropdown menu)

Buttons include 'Edit' (next to Result Type), 'Browse' (next to Icon), 'OK', 'Cancel', 'Apply', and 'Help'.

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.
2. A function activity must have an Internal Name (all uppercase and no 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.



Attention: You cannot update the internal name of an activity once it is defined.

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

3. Enter the name of the PL/SQL stored procedure you want this function activity to execute, using the format:
<package_name>.<procedure_name>. See: Standard API for PL/SQL Procedures Called by Function Activities: page 6 – 2.
4. Indicate the result type (a predefined Lookup Type) of 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 – 27.

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. 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: page 2 – 47.

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.

6. Specify the cost of this function activity. See: Activity Cost: page 3 – 43.
7. Choose Apply to save your changes.
8. Select the Details tab to display and modify the Details property page of the activity. See: To Define Optional Activity Details: page 3 – 50.
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 – 24.
11. If you externalize a parameter for the PL/SQL stored procedure that your function activity calls, you can expose that parameter in Oracle Workflow Builder as an attribute 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 – 19.

When you include a function activity in a process, you can assign a value to that function activity's attribute by displaying the Attribute Values properties page for that specific instance of the function activity. See: To Define Activity Attribute Values: page 4 – 9.

See Also

Using the Edit Button in a Property Page: page 3 – 6

► To Create a Process Activity

Before you can draw a workflow process diagram, you must first create a process activity in the navigator tree. Double-click on the process activity to display its Process window. You can then drag and drop activities from the navigator tree into the Process window to create your workflow diagram. See: Process Window: page 4 – 2.



The screenshot shows the 'Oracle Workflow Navigator Control Properties' dialog box with the 'Activity' tab selected. The dialog has four tabs: 'Activity', 'Details', 'Roles', and 'Access'. The 'Activity' tab contains the following fields and controls:

- Internal Name:** A text field containing 'REQUISITION_APPROVAL'.
- Display Name:** A text field containing 'Requisition Approval'.
- Description:** A text field containing 'Notify the appropriate manager(s) to approve a requisition an'.
- Result Type:** A dropdown menu set to 'Approval' with an 'Edit' button to its right.
- Runnable:** A checked checkbox.
- Icon:** A dropdown menu set to 'PROCESS.ICO' with a gear icon and a 'Browse' button to its right.

At the bottom of the dialog are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'.

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.
2. A process activity must have an Internal Name (all uppercase and no 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.



Attention: You cannot update the internal name of an activity once it is defined.

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

3. Indicate the result type (a predefined Lookup Type) of this activity. Result types list the possible results returned by this process. See: To Create Lookup Types: page 3 – 27.

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: page 2 – 47.

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 called by the Workflow Engine *CreateProcess* API and be run as an independent process. 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: page 7 – 9.
6. Choose Apply to save your changes.
7. Select the Details tab to display and modify the Details property page of the activity. See: To Define Optional Activity Details: page 3 – 50.
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 – 24.

See Also

Using the Edit Button in a Property Page: page 3 – 6

► **To Define Optional Activity Details**

The screenshot shows the 'Oracle Workflow Navigator Control Properties' dialog box with the 'Details' tab selected. The 'Error Process' field contains 'WFDEMO_ERROR_PROCESS'. The 'Timeout' section has three input fields: '0' for 'days', '0' for 'hrs', and '0' for 'mins'. The 'Effective' field is empty. The 'Loop Reset' checkbox is checked, and the 'Version' field contains '0'. At the bottom are buttons for 'OK', 'Cancel', 'Apply', and 'Help'.

1. If you are creating a process activity, specify the internal name of the error process activity to execute if an error occurs in the process. See: Default Error Process: page 5 – 16.
2. You can specify any combination of days, hours and minutes before a notification activity times out. If the notification activity is not completed by the specified time, you can redirect the process to transition to a different activity. See: Timeout Transitions: page 4 – 3.

3. The Effective date tells you when this version of the activity is available for the Workflow Engine to execute.

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. If Loop Reset is not checked, the activity can only be executed once, even if it is transitioned to more than once in a process.

If Loop Reset is checked, the activity is executed every time it is transitioned to, however, with every visit, the other activities that are a part of this loop (or transitioned to, after this activity) are reset. See: Looping: page 7 – 5.

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.
7. The Roles tab page is reserved for a future release.
8. Select the Access tab page to set the access levels allowed to modify this activity. See: Allowing Access to an Object: page 3 – 24.
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 activity now appears beneath the appropriate activity heading you defined it for 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.

► **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 activity or a notification activity also copies any attributes or message associated with it, respectively.

CHAPTER

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.

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. 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 activity to the next using the secondary mouse button.

Notification, function, 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 Approval Process Activities: page 10 – 10.

You can also create a <Default> 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 a <Timeout> transition if the notification activity times out before completion. See: Setting Up Background Workflow Engines: page 2 – 43.

Activities can also have multiple transitions for a single result, creating parallel branches.

Timeout Transitions

Draw a <Timeout> transition from one activity to another to force the process to perform another activity if the original notification activity is not completed by a specified period of time. See: To Define Optional Activity Details: page 3 – 50.

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 cancelled notification was expecting a response and the performer normally gets notifications via E-mail.

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 instead, the error process associated with the timed out activity or its parent process(es).

Note: You must have a background engine set up to process timed out activities. See: Setting Up Background Workflow Engines: page 2 – 43.

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, especially if there are results-based or parallel branches occurring further upstream from this node. In these cases, you want the activity 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 Loop Reset flag for an activity determines whether the activity (a function, notification or process) reexecutes when it is revisited more than once. Loop Reset is set in an activity's Details property page.

If Loop Reset is unchecked, the activity runs once and only once. If it is transitioned to a second time, the workflow engine ignores this activity. If Loop Reset is checked, the activity executes every time it is visited, however, with every visit, this activity as well as all other activities that

are a part of this loop are reset. 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: page 7 – 5 and Loop Counter Activity: page 5 – 5.



Suggestion: 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 – 2.

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 of the activity node's icon in the Process window.

Initiating a Process

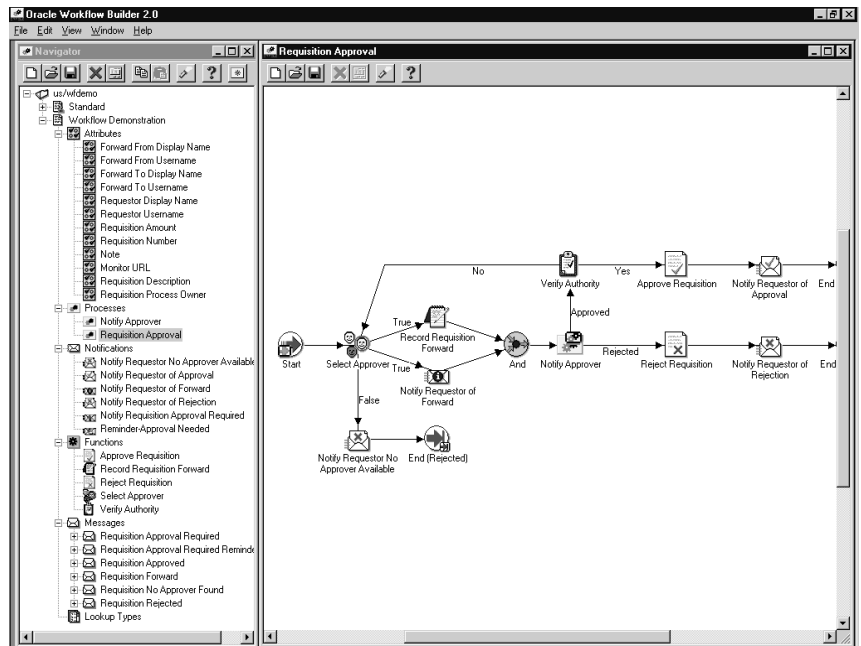
A workflow process begins when an application calls the Workflow Engine *CreateProcess()* and *StartProcess()* APIs. A subprocess is started when the Workflow Engine transitions to a process activity that represents the subprocess. See: Workflow Engine APIs: page 7 – 8.

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 – 5
- To Define Nodes: page 4 – 7
- To Define Activity Attribute Values: page 4 – 9
- To Create and Edit a Transition: page 4 – 10
- To Display a Process Overview: page 4 – 11
- To Print a Process: page 4 – 12
- To Copy a Process Diagram to the Clipboard: page 4 – 12
- To Validate a Process Definition: page 4 – 12

► **To Add Nodes to a Workflow Process**



1. Double-click on a process activity on the navigator tree, or select the process activity and choose Process Details from the Edit menu. A Process window opens with the name of your process in the window title.
2. Create a new node in a process by dragging and dropping a notification, function, 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. You can also

choose Create Activity from the right mouse button menu while your cursor is in the Process window to create a new activity node.

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 a file called 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 item type called Standard 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.

3. You can also create a new node using the New Activity mouse button menu option. An Activities property page appears for you to select the activity for this node. See: To Define Nodes in a Process: page 4 – 7.
4. 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.
5. 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 – 10.

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> or <Timeout> to define a transition to take if the activity returns a result that does not match the result of any other transition, 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.

6. 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.

See Also

- Process Window Toolbar: page 0 – 7

► To Define Nodes in a Process

The screenshot shows the 'Oracle Workflow Navigator Control Properties' dialog box. It has two tabs: 'Process Activity' and 'Attribute Values'. The 'Process Activity' tab is active. It contains the following fields and controls:

- Item Type:** A dropdown menu showing 'Workflow Demonstration' and an 'Edit' button.
- Notification:** A dropdown menu showing 'Notify Requestor No Approver Available' and an 'Edit' button.
- Label:** A text input field containing 'NTF_REQUESTOR_NO_APPROVER'.
- Start/End:** A dropdown menu showing 'Normal'.
- Comment:** An empty text input field.
- Performer:** A section containing:
 - Type:** A dropdown menu showing 'Item Attribute'.
 - Requestor Username:** A dropdown menu showing 'Requestor Username' and an 'Edit' button.

At the bottom of the dialog are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'.

1. If you use the right mouse button menu option New Activity in the Process window to create a new node, the property pages for the node appears. Select the name of the item type and activity in the Activities property page. If you create a node by dragging and dropping an activity from the navigator tree into the process window, then double-click on the node to display the property pages so you can further specify the details of the node.
2. 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 'Nth' instance of the activity in use.



Attention: Most APIs pass the activity's label to the workflow engine and not the activity name. See: Workflow Engine APIs: page 7 – 8.

3. 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.

4. 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 derive from the lookup type defined as the process activity's result type.
5. You can provide a comment to yourself about this node.
6. For notification activities, specify the performer of the activity, that is, the role to whom the notification is sent. You may either enter a role name or specify an item type attribute that dynamically determines a role at runtime. See: Roles: page 4 – 16.

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: page 7 – 77.

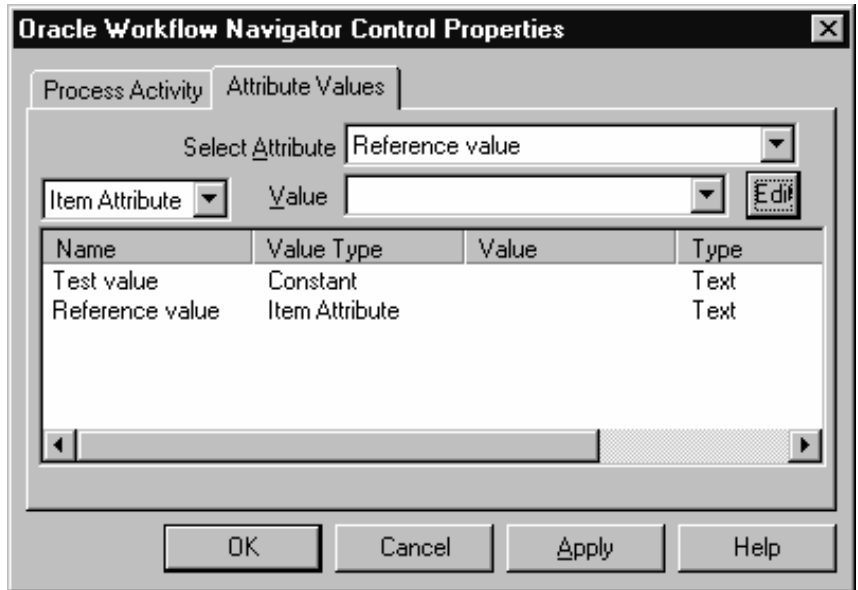
7. Choose Apply to save your changes.
8. If the node is a function activity and the function activity has activity attributes, you can assign values to those activity attributes by choosing the Attribute Values tab to display the Attribute Values property page.
9. If the node is a process activity, then 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.
10. You should 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.

See Also

- To Find an Object in the Navigator Tree: page 3 – 5

► **To Define Activity Attribute Values**

Function activity attribute values are used by the PL/SQL stored procedure that the function activity calls. See: To Define an Item Type or Activity Attribute: page 3 – 19.



1. Double-click on an activity node in the Process window to display its properties. Display the Attribute Values property page.
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.

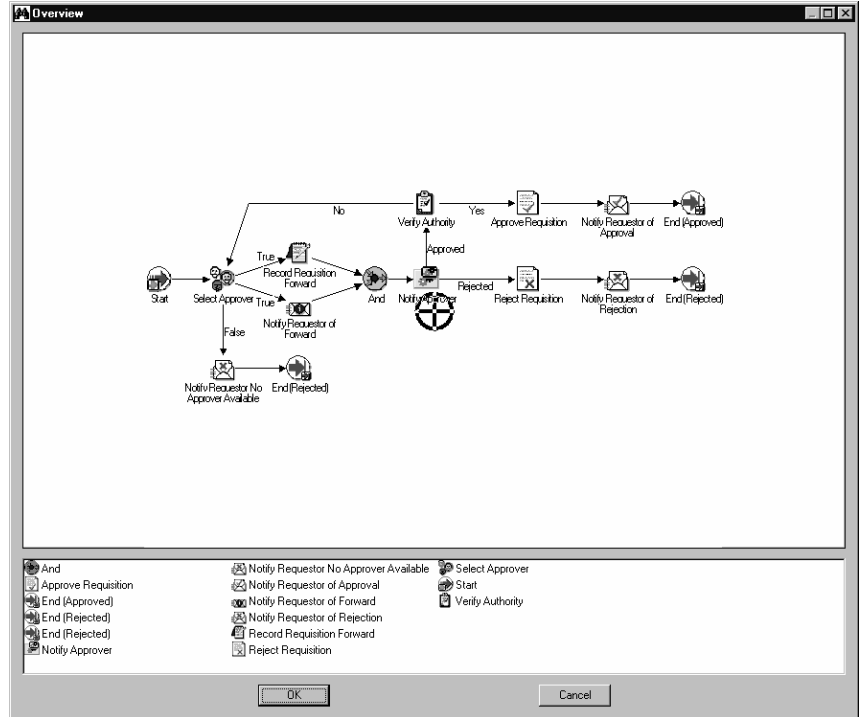
The value you enter must match the data type of the activity attribute, and for the actual activity parameter itself as it is defined in the PL/SQL function associated with the activity. The attribute type is displayed along with the name, description, value type, and value of each attribute in the attributes 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.
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. 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.
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 being 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 than 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:

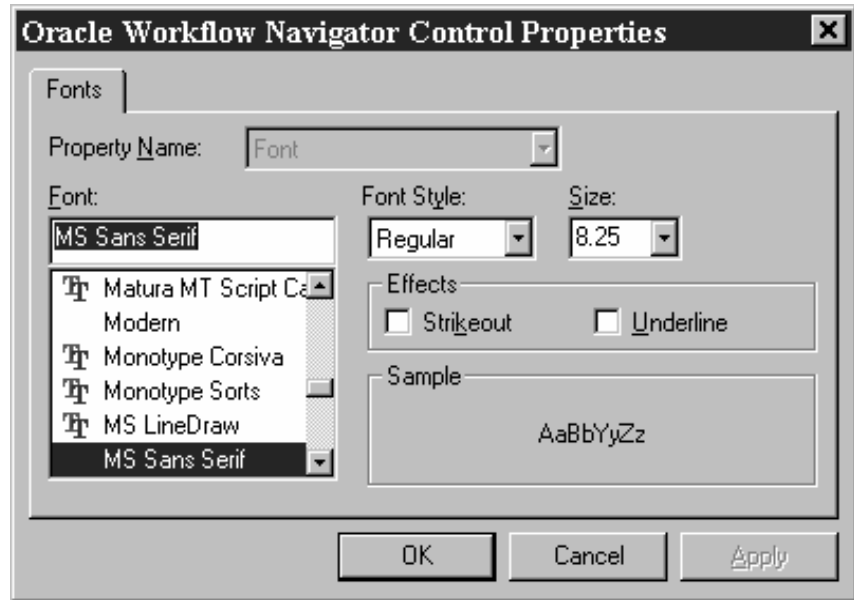
- Checks that a process has at least one Start and one End activity.
- Verifies that a process does not contain itself as a process activity.
- Validates that all possible activity results are modelled as outgoing transitions. If an activity completes with a result that is not associated with an outgoing 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.



Attention: 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.

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 properties 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 95 or Windows NT 4.0 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.

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 reference the roles stored in a database.

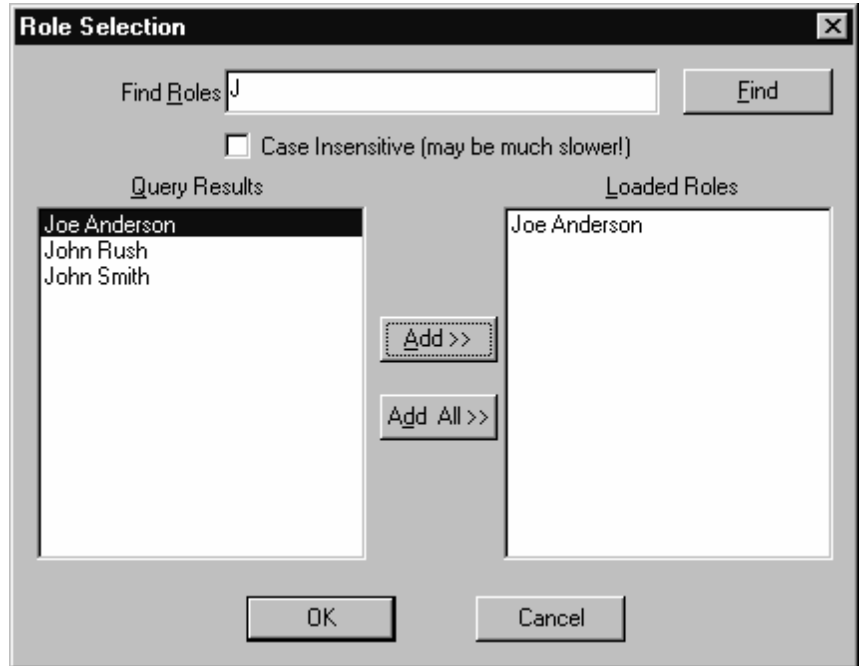
One example of how roles are referenced in Oracle Workflow Builder is when you include a notification activity in a process. You must assign that activity to a performer. The performer can be a designated role or an item type attribute that dynamically returns a role. To assign a performer to a role, 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: page 2 – 7* and *To Define Nodes in a Process: page 4 – 7*.

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.

► 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 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.

Enter standard SQL query syntax as search criteria in the Find Roles field. The field automatically appends % to the end of any search criteria.



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 objects in your open item type.

The 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, as shown in the following Process Activity 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 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.

Oracle Workflow Navigator Control Properties [X]

Role

Internal Name: DMCKEE

Display Name: David McKee

Description:

Language: AMERICAN Territory: AMERICA

Email Address: DMCKEE

Fax Number:

Notification Preference: Mail with HTML [v]

OK Cancel Apply Help

CHAPTER

5

Predefined Workflow Activities

This chapter tells you how to use Oracle Workflow's predefined 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 Workflow 2.0 *data* subdirectory.

Note: Predefined activities are also available for the predefined workflows shipped with Oracle Applications and Oracle Self-Service Web Applications. For more information on Oracle Applications-specific workflow activities, consult the documentation or help for that specific Oracle Application 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.

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 <code>WF_STANDARD.ANDJOIN</code> .
Or	Completes when the activities from at least one converging branch complete. Calls a PL/SQL procedure named <code>WF_STANDARD.ORJOIN</code> .

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. The first activity attribute requires as its value, a item type attribute of type Number, Date, or Text. The second activity attribute takes as its value, a constant number, date, or text string, respectively. See: *To Define Activity Attribute Values: page 4 – 9*.

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.

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*.

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 Date, enter a relative time expressed in *<days>.<fraction of days>*.
- **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 – 9.

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*: page 7 – 28

This activity calls the PL/SQL procedure named *WF_STANDARD.BLOCK*.

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.

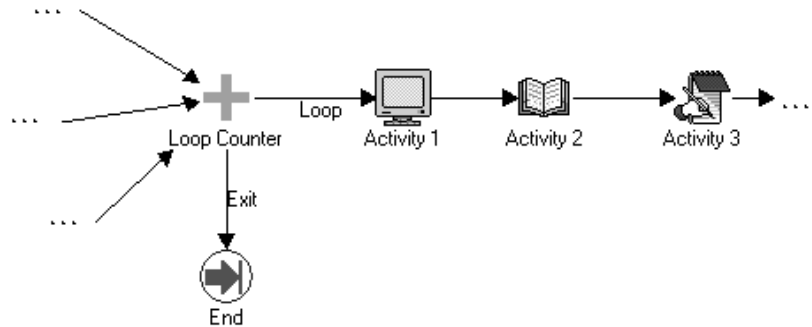
This Loop Counter activity calls the PL/SQL procedure named *WF_STANDARD.LOOPCOUNTER*.

Activity Attribute

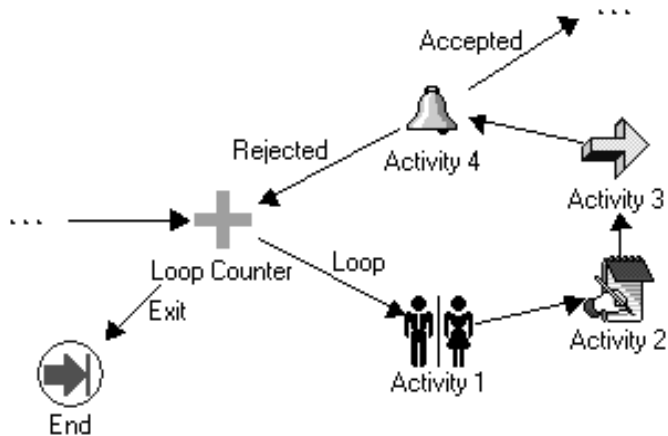
The Loop Counter activity has an activity attribute called Loop Limit. If the number of times that the Workflow Engine transitions to the Loop Counter activity is less than the value specified in Loop Limit, the Loop Counter activity will complete with a result of Loop and the engine will take the 'Loop' 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.



In a similar 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. See: To Define Activity Attribute Values: page 4 – 9.



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 – 9.

Voting Activity

The Voting activity 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.

In general, an activity can either send a notification message or perform a PL/SQL function. The special Voting activity, classified as a notification activity, does both by first sending a notification message to a group of users and then performing a PL/SQL function to tally the users' responses (votes).

The following four fields in the Activity properties page of the Voting activity determine how the Voting activity behaves:

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

Message Field

This field lists Default Voting Message as the message that the activity sends. The Default Voting Message includes a Respond message attribute with an Internal Name called RESULT. This attribute prompts the recipient to reply to the message using one of the possible responses listed in the lookup type specified.

You can edit the Default Voting Message or use your own custom message as long as you associate a Respond message attribute called RESULT with the message. Similarly, you can customize or use a different lookup type associated with RESULT to offer other possible responses to the recipient.

Result Type Field

This field contains the lookup type that lists the possible responses that the Voting activity would tally. You can update this field to use any lookup type as long as the lookup type matches the lookup type specified for the Respond message attribute called RESULT. The default lookup type is Yes/No.

Expand Roles Check Box

When Expand Roles is checked, an additional Function field appears. A checked value tells the Workflow Engine to poll for responses from

the multiple users in a role rather than just from the first user that replies, as is typically the case.



Attention: The Expand Roles Check Box should also be checked when sending FYI Notifications.

Function Field

This field lists the function that is used to tally the responses from users. The Voting activity calls the PL/SQL procedure *WF_STANDARD.VOTEFORRESULTTYPE* as the default function.

WF_STANDARD.VOTEFORRESULTTYPE is a generic tallying function. The Result Type that you specify for the activity defines the possible responses for the function to tally and the activity attributes you define determine how the function tallies the responses.

Activity Attributes

When you use the Voting activity, you must define a custom activity attribute of type Number for each possible voting response. Note that each possible voting response is actually 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 <Default> transition from the Voting activity exists, then the Workflow Engine will take this transition, otherwise, it will raise 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 <Default> transition from the Voting activity exists, then the

Workflow Engine will take this transition, otherwise, it will raise an error that no transition for the result is available (ERROR:#NOTTRANSITION).

Note: If you have something specific in mind that you would like to happen if a tie or no match occurs in a Voting activity, you can model that into a custom activity and create a <Default> transition from the Voting activity to that custom activity.

In addition to defining your set of custom activity attributes, you must also set an activity attribute called Voting Option, whose internal name must be VOTING_OPTION, to specify how the votes are tallied. You can set Voting Option to 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.

Example Voting Methods

1. Simple Majority

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

Table 5 - 1 (Page 1 of 1)

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

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

Table 5 – 2 (Page 1 of 1)

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

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

Table 5 – 3 (Page 1 of 1)

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

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

Table 5 - 4 (Page 1 of 1)

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

5. Black Ball

Response	Custom Response Activity Attribute Value
YES	100
NO	blank

Table 5 - 5 (Page 1 of 1)

Any vote for response NO makes NO the result.

6. Jury

Response	Custom Response Activity Attribute Value
GUILTY	100
NOT_GUILTY	100

Table 5 - 6 (Page 1 of 1)

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

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 each detail process reaches a certain point in its flow or vice versa.

When you spawn a detail process from a master process in Oracle Workflow, you need to define the master/detail relationship between the two processes by making a call to the workflow Engine *SetItemParent* API when you create the detail process. See: *SetItemParent*: page 7 – 33.

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*.

Activity Attributes

Use the "Continuation Flow" activity attribute to specify whether this activity is waiting for a corresponding "Master" or "Detail" process to complete. Use the "Continuation Activity" activity attribute to specify the label of the activity node that must complete in the corresponding process before the current process continues. The default value for the "Continuation Activity" activity attribute is *CONTINUEFLOW*. You can specify the value of the activity attribute in the Process window. See: *To Define Activity Attribute Values*: page 4 – 9.

Continue Flow

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*.

Activity Attributes

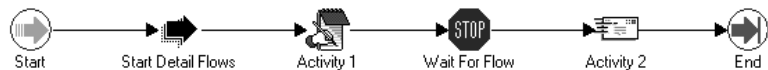
Use the "Waiting Flow" activity attribute to specify whether the halted process that is waiting for this activity to complete is a "Master" or "Detail" flow. Use the "Waiting Activity" activity attribute to specify

the label of the activity node in the halted process that is waiting for this activity to complete. You can specify the value of the activity attribute in the Process window. See: To Define Activity Attribute Values: page 4 – 9.

Example

The following figures show an example of how these coordination activities can be used. In the master process below, 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. An example of one of the detail processes below shows that when the detail process begins, 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.

Master Process



Detail Process



Note: You can include a Wait for Flow activity in a master process without using a Continue Flow activity in one or more of its corresponding detail process. The Workflow Engine simply continues the master process as soon as all the other detail processes that do contain a Continue Flow activity complete the Continue Flow activity.

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.



Attention: 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

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

Use the activity attribute called Item Attribute to choose the name of the item attribute that you want to use to store the URL for the Workflow Monitor window. Use the activity attribute called Administration Mode to 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.

Default Error Process

Oracle Workflow includes an item type designed specifically for error processing called System: Error. To view the details of this item type, choose Open from the File menu, then connect to the database and select the System: Error item type or connect to a file called wferror.wft in the Workflow 2.0 *data* subdirectory. The System: Error item type has a list of item type attributes and the following components associated with it:

- **Default Error Process**—a process that notifies a performer that an error has occurred. If you edit this process, you see that it includes two activities: Default Error Notification and End Error Process.
- **End Error Process function activity**—a function activity that represents the end activity of the error process.
- **Reset Error**—a function activity that allows you to rerun the originally errored activity again after the error process completes. This activity includes two activity attributes, COMMAND and RESULT. The possible values for COMMAND are "Retry", "Reset", or "Skip" to rerun, reset, or skip the activity, respectively. The value for RESULT is the result you want to assign to the error activity if you set COMMAND to "Skip". This activity can be added to the Default Error Process or to any custom error process. See: To Define Activity Attribute Values: page 4 – 9.
- **Default Error Notification**—a notification activity that sends a message called Workflow Default Error Message. The message includes the following 'Send' message attributes to provide information about the activity in error:
 - Error Item Type
 - Error Item Key
 - Error Name
 - Error Message
 - Error Stack
 - Error Activity ID
 - Error Activity Label
(of the format *<process_name>:<instance_label>*)
 - Error Result Code
 - Error Notification ID

– Error Assigned User

Oracle Workflow allows you to customize the Default Error Process to handle errors in your workflow process. You can also create your own custom error processes using the activities associated with the System: Error item type. Once you define an error process, you can assign it to any process or activity. See: To Define Optional Activity Details: page 3 – 50.

Note: Rather than use the Default Error Process, you should try to handle errors due to business rule incompatibilities by modelling 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 the Default Error Process, the entire workflow process will have an 'Error' status and will halt until a workflow administrator handles the error.

See Also

Workflow Core APIs: page 7 – 34

CHAPTER

6

Defining PL/SQL Procedures for Oracle Workflow

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

Standard API for PL/SQL Procedures Called by Function Activities

All PL/SQL stored procedures that are called by function activities in an Oracle Workflow process should follow a standard API format. You should also use this API when you define any custom PL/SQL stored procedures for your workflow processes so that the Workflow Engine can properly execute your activity.

The example in this section is numbered with the notation 1→ for easy referencing. The numbers and arrows themselves are not part of the procedure.

```
1→ procedure <procedure name>          ( itemtype    in varchar2,
                                         itemkey      in varchar2,
                                         actid        in number,
                                         funcmode    in varchar2,
                                         result      out varchar2 ) is
2→   <local declarations>
3→   begin
         if ( funcmode = 'RUN' ) then
             <your RUN executable statements>
             resultout := 'COMPLETE:<result>';
             return;
         endif;
4→   if ( funcmode = 'CANCEL' ) then
             <your CANCEL executable statements>
             resultout := 'COMPLETE';
             return;
         end if;
5→   if ( funcmode = '<other funcmode>' ) then
             resultout := ' ';
             return;
         endif;
6→   exception
         when others then
             WF_CORE.CONTEXT ('<package name>', '<procedure name>', <itemtype>, <itemkey>,
                               to_char(<actid>), <funcmode>);
             raise;
7→   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 that this procedure is called from.
funcmode	The execution mode of the function activity. Either 'RUN', 'CANCEL', or 'TIMEOUT'. Other execution modes may be added in the future.
result	<p>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:</p> <p>COMPLETE:<i><result_code></i>—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.</p> <p>WAITING—activity is set to wait a specified period of time before it completes. A background engine continually checks for when the activity wait period completes, at which point it processes the activity as complete.</p> <p>DEFERRED:<i><date></i>—activity is deferred to a background engine for execution until a given date. <i><date></i> must be of the format: <code>to_char(<date_string>, wf_engine.date_format)</code></p> <p>NOTIFIED:<i><notification_id></i>:<i><assigned_user></i>—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 <code>CompleteActivity()</code> to inform the Workflow engine when the action completes.</p>

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'`.

4→ This section clears the activity and can contain executable statements that run if the value of `funcmode` is `'CANCEL'`. Often times, 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 these special cases:

- The activity is part of a loop that is being revisited.

The first activity in a loop must always have the Loop Reset flag checked in the Activities properties Detail page. When the Workflow Engine encounters an activity that has already run, it verifies whether the activity's Loop Reset flag is set. If the flag is set, 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 reverse order and executes each activity in `'CANCEL'` mode to clear all prior results for the activities so they can run again. See: [Looping: page 7 – 5](#) and [Loop Counter Activity: page 5 – 5](#).

- The activity is part of a process that has been cancelled by a call to the `AbortProcess` Workflow Engine API. You may want to cancel a process to clear its current results. See: [AbortProcess: page 7 – 18](#).

5→ This section handles execution modes other than `'RUN'` or `'CANCEL'`. Currently `'TIMEOUT'` is the only other execution mode available, but others 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.

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. See: [CONTEXT: page 7 – 39](#).

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 that a function activity in an item type will need to execute, just before it executes a function activity in a new database session.
- **TEST_CTX**—to determine if a form can be launched with the current context information just before the Oracle Applications Notification Viewer form launches a reference form. If the context is incorrect, the form cannot be launched and a message is displayed to that effect. See: *Reviewing Notifications in the Notification Viewer Form*: page 8 – 2.

The standard API for the selector/callback function is as follows. This section is numbered with the notation 1→ for easy referencing. The numbers and arrows 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,
			result	in out varchar2) is
2→	<local declarations>			
3→	begin			
	if (command = 'RUN') then			
	<your RUN executable statements>			
	resultout := '<Name of process to run>';			
	return;			
	endif;			
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> ';			
	return;			
	endif;			
6→	if (command = '<other command>') then			
	resultout := ' ';			
	return;			
	endif;			
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.
result	<p>A result may be returned depending on the command that is used to call the selector/callback function.</p> <p>If the function is called with 'RUN', the name of the process to run must be returned through the result 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 or 'FALSE' if the context is incorrect. If any other value is returned, Oracle Workflow assumes that this command is not implemented by the callback.</p>

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 a 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 just before it begins to execute a function activity in a new database session. This command is useful when you need to set item type context 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 when the Oracle Applications Notification Viewer form is just about to launch a reference form. The code in this section should return `'TRUE'` if the context is correct, and `'FALSE'` if the context is incorrect. If the context is incorrect, you can raise an exception and place a message in the `WF_CORE` error system to indicate the reason the context is invalid. The raised exception is also printed in an error message in the form.

5→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.

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. See: `CONTEXT`: page 7 – 39.

Standard API for a "PL/SQL" Document

You can integrate a document into a workflow process by defining the document as an attribute for an item type, message, or activity. One type of document that Oracle Workflow supports is a "PL/SQL" document. Oracle Workflow constructs a dynamic call to a PL/SQL procedure that generates the document. The PL/SQL procedure must have the following standard API:

procedure <i><procedure name></i>	(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 document identifier that you specify in the default value field of the Attribute property page for a "PL/SQL" document (plsql:*<procedure>*/*<document_identifier>*).

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 email 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*; page 2 – 36.

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 a 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. 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.
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.

CHAPTER

7

Oracle Workflow APIs

This chapter describes the APIs for Oracle Workflow. The APIs consists of views and PL/SQL functions and procedures that you can use to access the Workflow Engine, the Notification System, and workflow data.

Oracle Workflow Procedures and Functions

Oracle Workflow supplies a list of public PL/SQL procedures and functions that you can use to set up a workflow process. They are grouped within the following packages:

- WF_ENGINE: page 7 – 8
- WF_CORE: page 7 – 34
- WF_PURGE: page 7 – 42
- WF_DIRECTORY: page 7 – 49
- WF_MONITOR: page 7 – 58
- Oracle Workflow Views: page 7 – 62
- WF_NOTIFICATIONS: page 7 – 67

Overview of the Workflow Engine

The Workflow Engine manages all automated aspects of a workflow process for each item. The engine is implemented in server-side PL/SQL and is activated whenever a call to a workflow procedure or function is made.

Additionally, Workflow Engines can be set up as background tasks to perform activities that are too costly to execute in real time.

The Workflow Engine performs the following services for a client application:

- It manages the state of all activities for an item, and in particular, determines which new activity to transition to whenever a prerequisite activity completes.
- It automatically executes function activities (execution is either immediate or else is deferred to a background engine) and sends notifications.
- It maintains a history of an activity's status.
- It detects error conditions and executes error processes.

The state of a workflow item is defined by the various states of all activities that are part of the process for that item. The engine changes activity states in response to an API call to update the activity family of calls. The API calls that update activity states are:

- CreateProcess
- StartProcess
- CompleteActivity
- AssignActivity
- HandleError
- SuspendProcess
- ResumeProcess
- AbortProcess

Based on the result of a previous activity, the engine also attempts to execute the next activity directly. An activity may have the following status:

- Active
- Complete
- Waiting

- Notified
- Deferred
- Error
- Suspended

Completion Processing

Engine processing is triggered whenever a process activity completes and calls the Workflow Engine API. The engine then attempts to execute (or mark for deferred execution) all activities that are dependent on the completed activity.

Note: A process as a whole can complete but still contain activities that were visited but not yet completed. For example, a completed process may contain a standard Wait activity that is not complete because the designated length of time to wait has not yet elapsed. When the process as a whole completes, the Workflow Engine marks these incomplete activities as having a status of COMPLETE and a result of #FORCE. This distinction is important if you need to include this information in your process status reports.

Deferred Processing

The engine has a deferred processing feature that allows long-running tasks to be handled by background engines instead of in real time. Deferring the execution of activity functions to background engines allows the Workflow Engine to move forward to process other activities that are currently active. The engine can be set up to operate anywhere on a continuum between processing all eligible work immediately, to processing nothing and marking all transitions as deferred.

Associated with each activity is a user-defined processing cost. This cost may be small if the activity merely sets an item attribute, or it may be very high if the activity performs a resource-intensive operation. Since the engine operates within the *CompleteActivity* procedure call, its execution is synchronous, meaning the engine must finish processing before control returns to the caller (for example, a form). If the result of a completed activity triggers the execution of a costly function, the caller might want to defer the execution of that costly function to a background engine.

To implement the deferred behavior, the engine recognizes a threshold cost above which it will simply mark an activity as DEFERRED rather

than attempting to execute it. A background engine can be set up to poll for deferred activities and to process them. Some sites may have multiple background engines operating at different thresholds, so that there would be no chance of tying up all background processing with long operations.

Error Processing

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). You can use Oracle Workflow Builder to define the events you want to occur in case of an error. Use Oracle Workflow Builder to modify the Default Error Process associated with the System:Error item type or create your own custom error process. See: Default Error Process: page 5 – 16.

The error process can include branches based on error codes, send notifications, and attempt to deal with the error using automated rules for resetting, retrying, or skipping the failed activity. Once you define an error process, you can associate it with any activity. The error process is then initiated whenever an error occurs for that activity. See: To Define Optional Activity Details: page 3 – 50.

The Workflow Engine traps errors produced by function activities by setting a savepoint before each function activity. If an activity produces an unhandled exception, or returns an invalid result, 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. In general, 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.

Looping

Looping occurs when the completion of an activity causes a transition to another activity that has already been completed. This can be handled in two ways: 1) Ignore the transition (a "run once" activity) or 2) Reset the process to the loop point (undo the activities within the loop). Every activity has a Loop Reset check box in Oracle Workflow

Builder so the you can specify the behavior for the activity if a loop occurs.

Turning Loop Reset off is useful for implementing activities that should only run once, even though they can be transitioned to from multiple sources. For example, this mode allows the implementation of a "logical OR" activity, which is transitioned to multiple times, but completes after the first transition. This mode would only be useful in user defined activities when implementing some other OR-like function.

Turning Loop Reset on for an activity causes the engine, upon transitioning to that completed activity, to transition backwards through the completed loop and reset that part of the process so that the engine can begin execution again from the reset point. An example of this behavior is if a process contained a loop where a purchase order were returned to the requester for modification and resubmission.

The engine clears the activities in a loop by:

- Building a list of all activities visited following the loop reset point, and
- Traversing the list in reverse order, cancelling each activity and resetting its status.

Cancelling an activity is similar to running the activity, except that the activity is executed in "CANCEL" mode instead of in "RUN" mode. The activity can then reverse any operation it performed to allow for reexecuting the loop. For example, if an activity sent out a notification, the notification would be canceled.

Version / Effective Date

Activities and processes are marked with a version number so that more than one version of an activity or a process definition can be in use at any one time. An effective date controls which version of the definition the engine uses when executing a process (the engine uses the latest eligible version). When you edit a process and save it to the database, you can save it with an immediate or future effective date. The version number is then increased by one. See: Retrieving and Saving Item Types: page 3 – 8.

Item Type Attributes

A set of item type attributes is defined at both design-time and runtime for each item. These attributes provide information to the function and

notification activities used in the processes associated with the item type.

Workflow Engine APIs

The following APIs can be called by an application program or a workflow function in the runtime phase to communicate with the engine and to change the status of each of the activities. These APIs are defined in a PL/SQL package called WF_ENGINE.

See Also

Standard API for PL/SQL Procedures Called by a Function Activities:
page 6 – 2

CreateProcess

Syntax `procedure CreateProcess`
 `(itemtype in varchar2,`
 `itemkey in varchar2,`
 `process in varchar2 default '');`

Description Creates a new runtime process for an application item.

For example, a Requisition item type may have a Requisition Approval Process as a top level process. When a particular requisition is created, an application calls CreateProcess to set up the information needed to start the defined process.

Arguments (input)	itemtype	A valid item type. Item types are defined in the Workflow Builder.
	itemkey	A string derived from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the new process and must be passed to all subsequent API calls for that process.
	process	An optional argument that allows the selection of a particular process for that item. Provide the process internal name. If process is null, the item type's selector function is used to determine the top level process to run. This argument defaults to null.

Caution: Although you can make a call to *CreateProcess()* and *StartProcess()* from a database trigger to initiate a workflow process, you should avoid doing so in certain circumstances. For example, if a database entity has headers, lines and details, and you initiate a workflow process from an AFTER INSERT trigger at the header-level of that entity, your workflow process may fail because some subsequent activity in the process may require information from the entity's lines or details level that is not yet populated.

Caution: The Workflow Engine always issues a savepoint before executing each activity so that it can rollback to the previous activity in case an error occurs. Because of this feature, you should avoid initiating a workflow process from a database trigger because savepoints and rollbacks are not allowed in a database trigger.

If you must initiate a workflow process from a database trigger, you must immediately defer the initial start activities to a background engine, so that they are no longer executing from a database trigger. To accomplish this:

- Set the cost of the process start activities to a value greater than the Workflow Engine threshold (default value is 0.05 second).

or

- Set the Workflow Engine threshold to be less than 0 before initiating the process:

```
begin
    WF_ENGINE.threshold := -1;
    WF_ENGINE.CreateProcess(...);
    WF_ENGINE.StartProcess(...);
end
```

(This method has the same effect as the previous method, but is more secure as the initial start activities are always deferred even if the activities' costs change.

SetItemUserKey

Syntax

```
procedure SetItemUserKey
    (itemtype in varchar2,
     itemkey in varchar2,
     userkey in varchar2);
```

Description Lets you set a user-friendly identifier for an item in a process, which is initially identified by an item type and item key. The user key is intended to be a user-friendly identifier to locate items in the Workflow Monitor and other components of Oracle Workflow.

Arguments (input)	itemtype	A valid item type.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the item in a process. See: CreateProcess : page 7 – 9.
	userkey	The user key to assign to the item identified by the specified item type and item key.

GetActivityLabel

Syntax `function GetActivityLabel`
 `(actid in number)`
 `return varchar2;`

Description Returns the instance label of an activity, given the internal activity instance ID. The label returned has the following format, which is suitable for passing to other Workflow Engine APIs, such as CompleteActivity and HandleError, that accept activity labels as arguments:

`<process_name>:<instance_label>`

Arguments (input) **actid** An activity instance ID.

SetItemOwner

Syntax `procedure SetItemOwner`
 `(itemtype in varchar2,`
 `itemkey in varchar2,`
 `owner in varchar2);`

Description A procedure to set the owner of existing items. The owner must be a valid role.

Arguments (input)	itemtype	A valid item type. Item types are defined in the Workflow Builder.
	itemkey	A string derived from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the new process and must be passed to all subsequent API calls for that process.
	owner	A valid role.

StartProcess

Syntax procedure StartProcess
 (itemtype in varchar2,
 itemkey in varchar2);

Description Begins execution of the specified process. The engine locates the activity marked as **START** and then executes it. *CreateProcess()* must first be called to define the itemtype and itemkey before calling *StartProcess()*.

Arguments (input)

itemtype	A valid item type.
itemkey	A string derived from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: <i>CreateProcess</i> : page 7 – 9.

Caution: Although you can make a call to *CreateProcess()* and *StartProcess()* from a trigger to initiate a workflow process, you should avoid doing so in certain circumstances. For example, if a database entity has headers, lines and details, and you initiate a workflow process from an **AFTER INSERT** trigger at the header-level of that entity, your workflow process may fail because some subsequent activity in the process may require information from the entity's lines or details level that is not yet populated.

Caution: The Workflow Engine always issues a savepoint before executing each activity so that it can rollback to the previous activity in case an error occurs. Because of this feature, you should avoid initiating a workflow process from a database trigger because savepoints and rollbacks are not allowed in a database trigger.

If you must initiate a workflow process from a database trigger, you must immediately defer the initial start activities to a background engine, so that they are no longer executing from a database trigger. To accomplish this:

- Set the cost of the process start activities to a value greater than the Workflow Engine threshold (default value is 0.05 second).

or

- Set the Workflow Engine threshold to be less than 0 before initiating the process:

```
begin
    WF_ENGINE.threshold := -1;
    WF_ENGINE.CreateProcess(...);
    WF_ENGINE.StartProcess(...);
end
```

(This method has the same effect as the previous method, but is more secure as the initial start activities are always deferred even if the activities' costs change.

SuspendProcess

Syntax procedure SuspendProcess
 (itemtype in varchar2,
 itemkey in varchar2,
 process in varchar2 default '');

Description Suspends process execution so that no new transitions occur. Outstanding notifications can complete by calling *CompleteActivity()*, but the workflow does not transition to the next activity. Restart suspended processes by calling *ResumeProcess()*.

Arguments (input)

itemtype	A valid item type.
itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: <i>CreateProcess</i> : page 7 – 9.
process	An optional argument that allows the selection of a particular subprocess for that item. Provide the process activity's label name. If the process activity label name does not uniquely identify the subprocess you can precede the label name with the internal name of its parent process. For example, <i><parent_process_internal_name>:<label_name></i> . If this argument is null, the top level process for the item is suspended. This argument defaults to null.

ResumeProcess

Syntax

```
procedure ResumeProcess
    (itemtype in varchar2,
     itemkey in varchar2,
     process in varchar2 default '');
```

Description Returns a suspended process to normal execution status. Any activities that were transitioned to while the process was suspended are now executed.

Arguments (input)

itemtype	A valid item type.
itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: CreateProcess: page 7 – 9.
process	An optional argument that allows the selection of a particular subprocess for that item type. Provide the process activity's label name. If the process activity label name does not uniquely identify the subprocess you can precede the label name with the internal name of its parent process. For example, <code><parent_process_internal_name>:<label_name></code> . If this argument is null, the top level process for the item is resumed. This argument defaults to null.

AbortProcess

Syntax `procedure AbortProcess`
 `(itemtype in varchar2,`
 `itemkey in varchar2,`
 `process in varchar2 default '',`
 `result_code in varchar2 default eng_force);`

Description Aborts process execution and cancels outstanding notifications. The process is considered complete, with a status specified by the `result_code` argument.

Arguments (input)

itemtype	A valid item type.
itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: CreateProcess: page 7 – 9 .
process	An optional argument that allows the selection of a particular subprocess for that item type. Provide the process activity's label name. If the process activity label name does not uniquely identify the subprocess you can precede the label name with the internal name of its parent process. For example, <code><parent_process_internal_name>:<label_name></code> . If this argument is null, the top level process for the item is aborted. This argument defaults to null.
result_code	A status assigned to the aborted process. The result code must be one of the values defined in the process Result Type, or one of the following standard engine values: <code>eng_exception</code> <code>eng_timeout</code> <code>eng_force</code> <code>eng_mail</code> <code>eng_null</code> This argument defaults to "eng_force".

Background

Syntax procedure Background

```
(itemtype in varchar2,  
  minthreshold in number default null,  
  maxthreshold in number default null,  
  process_deferred in boolean default TRUE,  
  process_timeout in boolean default TRUE);
```

Description Runs a background engine for processing deferred and/or timed out activities using the parameters specified. The background engine executes all activities that satisfy the given arguments at the time that the background engine is invoked. This procedure does not remain running long term, so you must restart this procedure periodically. Any activities that are newly deferred or timed out after the current background engine starts is processed by the next background engine that is invoked. You may run a script called wfbkgchk.sql to get a list of the activities waiting to be processed by the next background engine run. See: Wfbkgchk.sql: page 11 – 3.

If you are using the standalone version of Oracle Workflow, you can use one of the sample background engine looping scripts described below or create your own script to make the background engine procedure loop indefinitely. If you are using the version of Oracle Workflow embedded in Oracle Applications, you can use the concurrent program version of this procedure and take advantage of the concurrent manager to schedule the background engine to run periodically. To Schedule Background Engines: page 2 – 44

Arguments (input)	itemtype	A valid item type. If the item type is null the Workflow engine will run for all item types.
	minthreshold	Optional minimum cost threshold for an activity that this background engine processes, in hundredths of a second. There is no minimum cost threshold if this parameter is null.
	maxthreshold	Optional maximum cost threshold for an activity that this background engine processes in hundredths of a second. There is no maximum cost threshold if this parameter is null.
	process_deferred	Specify TRUE or FALSE to indicate whether to run deferred processes. Defaults to TRUE.
	process_timeout	Specify TRUE or FALSE to indicate whether to run timed out processes. Defaults to TRUE.

Example Background Engine Looping Scripts

For the standalone version of Oracle Workflow you can use one of two example scripts to repeatedly run the background engine regularly.

The first example is a sql script stored in a file called *wfbkg.sql* and is available on your server in the Oracle Workflow *admin/sql* subdirectory. To run this script, go to the directory where the file is located and type the following command at your operating system prompt:

```
sqlplus <username/password> @wfbkg <min> <sec>
```

Replace *<username/password>* with the Oracle7 database account username and password where you want to run the background engine. Replace *<min>* with the number of minutes you want the background engine to run and replace *<sec>* with the number of seconds you want the background engine to sleep between calls.

The second example is a shell script stored in a file called *wfbkg.csh* and is available on your server in the Oracle Workflow *bin* subdirectory. To run this script, go to the directory where the file is located and type the following command at your operating system prompt:

```
wfbkg.csh <username/password>
```

Replace *<username/password>* with the Oracle7 database account username and password where you want to run the background engine.

AddItemAttr

Syntax `procedure AddItemAttr`
 `(itemtype in varchar2,`
 `itemkey in varchar2,`
 `aname in varchar2);`

Description Adds an empty item type attribute variable to the process. Although most item type attributes are defined at design time, developers can create new attributes at runtime for a specific process.

Arguments (input)	itemtype	A valid item type.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: CreateProcess: page 7 – 9.
	aname	The internal name of the item type attribute.

SetItemAttribute

Syntax

```
procedure SetItemAttrText
    (itemtype in varchar2,
     itemkey in varchar2,
     aname in varchar2,
     avalue in varchar2);

procedure SetItemAttrNumber
    (itemtype in varchar2,
     itemkey in varchar2,
     aname in varchar2,
     avalue in number);

procedure SetItemAttrDate
    (itemtype in varchar2,
     itemkey in varchar2,
     aname in varchar2,
     avalue in date);
```

Description Sets the value of an item type attribute in a process. Use the correct procedure for your attribute type. All attribute types except number and date use *SetItemAttrText*.

Arguments (input)	itemtype	A valid item type.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: <i>CreateProcess</i> : page 7 – 9.
	aname	The internal name of an item type attribute.
	avalue	The value for an item type attribute.

GetItemAttribute

Syntax

```
function GetItemAttrText
    (itemtype in varchar2,
     itemkey in varchar2,
     aname in varchar2) return varchar2;

function GetItemAttrNumber
    (itemtype in varchar2,
     itemkey in varchar2,
     aname in varchar2) return number;

function GetItemAttrDate
    (itemtype in varchar2,
     itemkey in varchar2,
     aname in varchar2) return date;
```

Description Returns the value of an item type attribute in a process. Use the correct function for your attribute type. All attribute types except number and date use *GetItemAttrText*.

Arguments (input)	itemtype	A valid item type.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: <i>CreateProcess</i> : page 7 – 9.
	aname	The internal name of an item type attribute.

GetItemAttrInfo

Syntax `procedure GetItemAttrInfo`
 `(itemtype in varchar2,`
 `aname in varchar2,`
 `atype out varchar2,`
 `subtype out varchar2,`
 `format out varchar2);`

Description Returns information about an item type attribute, such as its type and format, if any is specified. Subtype information is not available for item type attributes. If the attribute is used as a message attribute, then this procedure returns the message attribute subtype, that is, whether the message attribute has a source of 'SEND' or 'RESPOND'.

Arguments (input)

itemtype	A valid item type.
aname	The internal name of a item type attribute.

GetActivityAttrInfo

Syntax

```
procedure GetActivityAttrInfo
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number,
     aname in varchar2,
     atype out varchar2,
     subtype out varchar2,
     format out varchar2);
```

Description Returns information about an activity attribute, such as its type and format, if any is specified. This procedure currently does not return any subtype information for activity attributes.

Arguments (input)	itemtype	A valid item type.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: CreateProcess: page 7 – 9 .
	actid	The activity ID for a particular usage of an activity in a process definition.
	aname	The internal name of an activity attribute.

GetActivityAttribute

Syntax

```
function GetActivityAttrText
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number,
     aname in varchar2) return varchar2;

function GetActivityAttrNumber
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number,
     aname in varchar2) return number;

function GetActivityAttrDate
    (itemtype in varchar2,
     itemkey in varchar2,
     actid in number,
     aname in varchar2) return date;
```

Description Returns the value of an activity attribute in a process. Use the correct function for your attribute type. If the attribute is a Number or Date type, then the appropriate function translates the number/date value to a text-string representation using the attribute format.

Note: Use *GetActivityAttrText* for Form, URLs, lookups and document attribute types.

Arguments (input)	itemtype	A valid item type.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process. See: <i>CreateProcess</i> : page 7 - 9.
	actid	The activity ID for a particular usage of an activity in a process definition.
	aname	The internal name of an activity attribute.

BeginActivity

Syntax

```
procedure BeginActivity
    (itemtype in varchar2,
     itemkey in varchar2,
     activity in varchar2);
```

Description Determines if the specified activity can currently be performed on the process item and raises an exception if it cannot.

The CompleteActivity() procedure automatically performs this function as part of its validation. However, you can use BeginActivity to verify that the activity you intend to perform is currently allowed before actually calling it. See: CompleteActivity: page 7 – 28.

Arguments (input)	itemtype	A valid item type.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process.
	activity	The process activity to perform. Provide the process activity's label name. If the process activity label name does not uniquely identify the activity you can precede the label name with the internal name of its parent process. For example, <code><parent_process_internal_name>:<label_name></code> .

Example `/*Verify that a credit check can be performed on an order. If it is allowed, perform the credit check, then notify the Workflow Engine when the credit check completes.*/`

```
begin
    wf_engine.BeginActivity('ORDER',
        to_char(order_id), 'CREDIT_CHECK')
    OK = TRUE
exception
    when others then
        WF_CORE.Clear
        OK = FALSE
end;
if OK then
    -- perform activity --
    wf_engine.CompleteActivity('ORDER', to_char(order_id),
        'CREDIT_CHECK' :result_code);
endif;
```

CompleteActivity

Syntax procedure CompleteActivity

 (itemtype in varchar2,
 itemkey in varchar2,
 activity in varchar2,
 result_code in varchar2);

Description Notifies the workflow engine that the specified activity has been completed for a particular item. This procedure can be called for the following situations:

- **To create a new item**—Call CompleteActivity for a START activity to create a new item. START activities are designated as the beginning of a process in the Workflow Builder. The item type and key specified in this call must be passed to all subsequent calls that operate on this item.
- **To indicate a completed activity with an optional result**—This signals the Workflow Engine that an asynchronous activity has been completed. This procedure requires that the activity currently has a status of 'Active' or 'Notified'. An optional activity completion result can also be passed. The result can determine what transition the process takes next.

Arguments (input)	itemtype	A valid item type.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process.
	activity	The name of the process activity that is completed. Provide the process activity's label name. If the process activity label name does not uniquely identify the subprocess you can precede the label name with the internal name of its parent process. For example, <code><parent_process_internal_name>:<label_name></code> .
	result_code	An optional activity completion result. Possible values are determined by the process activity's Result Type, or one of the engine standard results. See: AbortProcess: page 7 – 18.

Example 1 `/*Complete the 'ENTER ORDER' activity for the 'ORDER' item type. The 'ENTER ORDER' activity allows creation of new items since it is the start of a workflow, so the item is created by this call as well.*/`
`wf_engine.CompleteActivity('ORDER', to_char(order.order_id),`
`'ENTER_ORDER', NULL);`

Example 2 `/*Complete the 'LEGAL REVIEW' activity with status 'APPROVED'. The item must already exist.*/`
`wf_engine.CompleteActivity('ORDER', '1003', 'LEGAL_REVIEW',`
`'APPROVED');`

Example 3 `/*Complete the BLOCK activity which is used in multiple subprocesses in parallel splits.*/`
`wf_engine.CompleteActivity('ORDER', '1003',`
`'ORDER_PROCESS:BLOCK-3',`
`'null');`

AssignActivity

Syntax

```
procedure AssignActivity
    (itemtype in varchar2,
     itemkey in varchar2,
     activity in varchar2,
     performer in varchar2);
```

Description Assigns or reassigns an activity to another performer. This procedure may be called before the activity is transitioned to. For example, a function activity earlier in the process may determine the performer of a later activity.

If a new user is assigned to a notification activity that already has an outstanding notification, the outstanding notification is canceled and a new notification is generated for the new user.

Arguments (input)	itemtype	A valid item type.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process.
	activity	The label name of the process activity. If the process activity label name does not uniquely identify the activity you can precede the label name with the internal name of its parent process. For example, <parent_process_internal_name>:<label_name>.
	performer	The name of the user who will perform the activity (the user who receives the notification). The name should be a role name from the Oracle Workflow directory services.

HandleError

Syntax

```
procedure HandleError
    (itemtype in varchar2,
     itemkey in varchar2,
     activity in varchar2,
     command in varchar2,
     result in varchar2);
```

Description This procedure can be called from an activity in an ERROR process to handle any process activity that has encountered an error.

You can also call this procedure for any arbitrary activity in a process, to rollback part of your process to that activity. The activity that you call this procedure with can have any status and does not have to have been executed. The activity can also be in a subprocess, if the process activity label is not unique within the process you may precede the activity label name with the internal name of its parent process. For example, `<parent_process_internal_name>:<label_name>`.

This procedure clears the activity specified and all activities following it that have already been transitioned to by reexecuting each activity in 'Cancel' mode, as in the case of loop reset. See: Looping: page 7 – 5. For an activity in the 'Error' state, there are no other executed activities following it, so the procedure simply clears the errored activity.

Once the activities are cleared, this procedure resets any parent processes of the specified activity to a status of 'Active', if they are not already active.

The procedure then handles the specified activity based on the command you provide: SKIP or RETRY.

Arguments (input)	item_type	A valid item type.
	item_key	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process.
	activity	The process activity that encountered the error or that you want to undo. Provide the label name of the process activity. If the process activity label name does not uniquely identify the subprocess you can precede the label name with the internal name of its parent process. For example, <code><parent_process_internal_name>:<label_name></code> .

command	<p>One of two commands that determine how to handle the process activity:</p> <p>SKIP—mark the activity as complete with the supplied result and continue execution of the process from that activity.</p> <p>RETRY—reexecute the activity and continue execution of the process from that activity.</p>
result	The result you wish to supply if the command is SKIP.

Note: An item's active date and the version number of the process that the item is transitioning through can never change once an item is created. Occasionally, however, you may want to use `HandleError` to manually make changes to your process for an existing item.

If the changes you make to a process are minor, you can use `HandleError` to manually push an item through activities that will error or redirect the item to take different transitions in the process.

If the changes you want to make to a process are extensive, then you need to perform at least the following steps:

- Abort the process by calling `WF_ENGINE.AbortProcess()`.
- Purge the existing item by calling `WF_ENGINE.Items()`.
- Revise the process.
- Recreate the item by calling `WF_ENGINE.CreateProcess()`.
- Restart the revised process at the appropriate activity by calling `WF_ENGINE.HandleError()`.

SetItemParent

Syntax

```
procedure SetItemParent
    (itemtype in varchar2,
     itemkey in varchar2,
     parent_itemtype in varchar2,
     parent_itemkey in varchar2,
     parent_context in varchar2);
```

Description Defines the parent/child relationship for a master process and a detail process. This API must be called by any detail process spawned from a master process to define the parent/child relationship between the two processes. You make a call to this API after you call the *CreateProcess* API, but before you call the *StartProcess* API for the detail process.

Arguments (input)	itemtype	A valid item type.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the child process.
	parent_itemtype	A valid item type for the parent process.
	parent_itemkey	A string generated from the application object's primary key to uniquely identify the item within the parent item type. The parent item type and key together identify the parent process.
	parent_context	Context information about the parent.

Workflow Core APIs

PL/SQL procedures called by function activities can use a set of core Oracle Workflow APIs to raise and catch errors.

When a PL/SQL procedure called by a function activity either raises an unhandled exception, or returns a result beginning with 'ERROR:', the Workflow Engine sets the function activity's status to ERROR and sets the columns ERROR_NAME, ERROR_MESSAGE, and ERROR_STACK in the table WF_ITEM_ACTIVITY_STATUSES to reflect the error.

The columns ERROR_NAME and ERROR_MESSAGE get set to either the values returned by a call to *WF_CORE.RAISE()*, or to the SQL error name and message if no call to *RAISE()* is found. The column ERROR_STACK gets set to the contents set by a call to *WF_CORE.CONTEXT()*, regardless of the error source.

Note: The columns ERROR_NAME, ERROR_MESSAGE, and ERROR_STACK are also defined as item type attributes for the System: Error predefined item type. You can reference from the error process that you associate with an activity, the information in these columns. See: Default Error Process: page 5 – 16.

The following APIs can be called by an application program or workflow function in the runtime phase to handle error processing. These APIs are stored in the PL/SQL package called WF_CORE.

See Also

Standard API for an Oracle Workflow PL/SQL Stored Procedure: page 6 – 2

CLEAR

Syntax `procedure CLEAR;`

Description Clears the error buffers.

See Also

`GET_ERROR`: page 7 – 36

GET_ERROR

Syntax

```
procedure GET_ERROR
    (err_name out varchar2,
     err_message out varchar2
     err_stack out varchar2);
```

Description Returns the name of a current error message and the token substituted error message. Also clears the error stack. Returns null if there is no current error.

Example 1 `/*Handle unexpected errors in your workflow code by raising WF_CORE exceptions. When calling any public Workflow API, include an exception handler to deal with unexpected errors.*/`

```
declare
    errname varchar2(30);
    errmsg varchar2(2000);
    errstack varchar2(32000);
begin
    ...
    Wf_Engine.CompleteActivity(itemtype, itemkey, activity,
    result_code);
    ...
exception
    when others then
        wf_core.get_error(err_name, err_msg, err_stack);
        if (err_name is not null) then
            wf_core.clear;
            -- Wf error occurred. Signal error as appropriate.
        else
            -- Not a wf error. Handle otherwise.
        end if;
end;
```

See Also

CLEAR: page 7 – 35

TOKEN

Syntax procedure TOKEN
(token_name in varchar2,
token_value in varchar2);

Description Defines an error token and substitutes it with a value. Calls to *TOKEN()* and *RAISE()* raise predefined errors for Oracle Workflow that are stored in the WF_RESOURCES table. The error messages contain tokens that need to be replaced with relevant values when the error message is raised. This is an alternative to raising PL/SQL standard exceptions or custom-defined exceptions.

Arguments (input)	token_name	Name of the token.
	token_value	Value to substitute for the token.

See Also

RAISE: page 7 – 38

CONTEXT: page 7 – 39

RAISE

Syntax procedure RAISE
 (name in varchar2);

Description Raises an exception to the caller by supplying a correct error number and token substituted message for the name of the error message provided.

Calls to *TOKEN()* and *RAISE()* raise predefined errors for Oracle Workflow that are stored in the WF_RESOURCES table. The error messages contain tokens that need to be replaced with relevant values when the error message is raised. This is an alternative to raising PL/SQL standard exceptions or custom-defined exceptions.

Error messages for Oracle Workflow are initially defined in message files (.msg). The message files are located in the *res/<language>* subdirectory of the Oracle Workflow server directory structure for the standalone version of Oracle Workflow or on your server in the *resource/<language>* subdirectory under \$FND_TOP for the Oracle Applications-embedded version of Oracle Workflow. During the installation of the Oracle Workflow server, a program called Workflow Resource Generator takes the designated message files and imports the messages into the WF_RESOURCES table.

Arguments (input) **name** Internal name of the error message as stored in the table WF_RESOURCES.

See Also

TOKEN: page 7 – 37

CONTEXT: page 7 – 39

To run the Workflow Resource Generator: page 2 – 15

CONTEXT

Syntax procedure CONTEXT

```
                  (pkg_name IN VARCHAR2,  
                  proc_name IN VARCHAR2,  
                  arg1      IN VARCHAR2 DEFAULT '*none*',  
                  arg2      IN VARCHAR2 DEFAULT '*none*',  
                  arg3      IN VARCHAR2 DEFAULT '*none*',  
                  arg4      IN VARCHAR2 DEFAULT '*none*',  
                  arg5      IN VARCHAR2 DEFAULT '*none*');
```

Description Adds an entry to the error stack to provide context information that helps locate the source of an error. Use this procedure with predefined errors raised by calls to *TOKEN()* and *RAISE()*, with custom-defined exceptions, or even without exceptions whenever an error condition is detected.

Arguments (input)	pkg_name	Name of the procedure package.
	proc_name	Procedure or function name.
	arg1	First IN argument.
	argn	nth IN argument.

Example 1 /*PL/SQL procedures called by function activities can use the WF_CORE APIs to raise and catch errors the same way the Workflow Engine does.*/

```
package My_Package is  
  
procedure MySubFunction(  
    arg1 in varchar2,  
    arg2 in varchar2)  
is  
    ...  
begin  
    if (<error condition>) then  
        Wf_Core.Token('ARG1', arg1);  
        Wf_Core.Token('ARG2', arg2);  
        Wf_Core.Raise('ERROR_NAME');  
    end if;  
    ...  
exception  
    when others then  
        Wf_Core.Context('My_Package', 'MySubFunction', arg1,
```

```

arg2);
    raise;
end MySubFunction;
procedure MyFunction(
    itemtype in varchar2,
    itemkey in varchar2,
    actid in number,
    funcmode in varchar2,
    result out varchar2)
is
...
begin
    ...
    begin
        MySubFunction(arg1, arg2);
    exception
        when others then
            if (Wf_Core.Error_Name = 'ERROR_NAME') then
                -- This is an error I wish to ignore.
                Wf_Core.Clear;
            else
                raise;
            end if;
        end;
    ...
exception
    when others then
        Wf_Core.Context('My_Package', 'MyFunction', itemtype,
            itemkey, to_char(actid), funcmode);
        raise;
end MyFunction;

```

See Also

TOKEN: page 7 – 37

RAISE: page 7 – 38

TRANSLATE

Syntax function TRANSLATE
 (tkn_name IN VARCHAR2)
 return VARCHAR2;

Description Translates the string value of a token.

Arguments (input) **tkn_name** Token name.

Workflow Purge APIs

The following APIs can be called by an application program or workflow function in the runtime phase to purge obsolete runtime data. These APIs are defined in the PL/SQL package called **WF_PURGE**.

WF_PURGE can be used to purge obsolete runtime data about completed items and processes, and also to purge information about obsolete activity versions that are no longer in use. You may want to periodically purge this obsolete data from your system to increase performance. The 3 most commonly used procedures are:

WF_PURGE.ITEMS – purge all runtime data associated with completed items, their processes, and notifications sent by them

WF_PURGE.ACTIVITIES – purge obsolete versions of activities that are no longer in use by any item.

WF_PURGE.TOTAL – purge both item data and activity data

The other auxiliary routines purge only certain tables or classes of data, and can be used in circumstances where a full purge is not desired.

See Also

Standard API for an Oracle Workflow PL/SQL Stored Procedure: page 6 – 2

Item_Activity_Statues

Syntax `procedure Item_Activity_Statues`
 `(itemtype in varchar2 default null,`
 `itemkey in varchar2,`
 `enddate in date default sysdate);`

Description Deletes from the tables WF_ITEM_ACTIVITY_STATUSES and WF_ITEM_ACTIVITY_STATUSES_H all rows of data associated with the specified item type and whose value for END_DATE is less than or equal to the specified end date.

Arguments (input)	itemtype	Item type associated with the activity statuses you want to delete. Leave this argument null to delete activity statuses for all item types.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. If null, the procedure purges all items in the specified itemtype.
	enddate	Specified date to delete up to.

Items

Syntax `procedure Items`
`(itemtype in varchar2 default null,`
`itemkey in varchar2,`
`enddate in date default sysdate);`

Description Deletes all items for the specified item type and end date using *Item_Activity_Statuses()*. Deletes from the tables WF_ITEM_ATTRIBUTE_VALUES and WF_ITEMS all rows of data associated with the specified item type and whose value for END_DATE is less than or equal to the specified end date.

Arguments (input)	itemtype	Item type to delete. Leave this argument null to delete all item types.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. If null, the procedure purges all items in the specified itemtype.
	enddate	Specified date to delete up to.

Notifications

Syntax procedure Notifications
 (itemtype in varchar2 default null,
 enddate in date default sysdate);

Description Deletes old notifications from the tables
WF_NOTIFICATION_ATTRIBUTES and WF_NOTIFICATIONS
associated with the specified item type, have an END_DATE less than
or equal to the specified end date and are not referenced by an existing
item.

Note: You should call *Items()* before calling *Notifications()* to
avoid having obsolete item references prevent obsolete
notifications from being deleted.

Arguments (input)	itemtype	Item type associated with the notifications you want to delete. Leave this argument null to delete notifications for all item types.
	enddate	Specified date to delete up to.

Item_Notifications

Syntax

```
procedure Item_Notifications
    (itemtype in varchar2 default null,
    itemkey in varchar2 default null,
    enddate in date default sysdate);
```

Description Deletes notifications sent by a particular item from the tables WF_NOTIFICATION_ATTRIBUTES and WF_NOTIFICATIONS associated with the specified item type, have an END_DATE less than or equal to the specified end date and are not referenced by an existing item.

Note: You should call *Items()* before calling *Item_Notifications()* to avoid having obsolete item references prevent obsolete notifications from being deleted.

Arguments (input)	itemtype	Item type associated with the notifications you want to delete. Leave this argument null to delete notifications for all item types.
	itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. If null, the procedure purges all items in the specified itemtype.
	enddate	Specified date to delete up to.

Total

Syntax procedure Total
 (itemtype in varchar2 default null,
 itemkey in varchar2,
 enddate in date default sysdate);

Description Deletes all obsolete runtime item type and activity data associated with the specified item type and have an END_DATE less than or equal to the specified end date.

Arguments (input)

itemtype	Item type associated with the obsolete runtime data you want to delete. Leave this argument null to delete obsolete runtime data for all item types.
itemkey	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. If null, the procedure purges all items in the specified itemtype.
enddate	Specified date to delete up to.

Workflow Directory Services APIs

The following APIs can be called by an application program or a workflow function in the runtime phase to retrieve information about the users and roles in the Oracle Workflow directory services. These APIs are defined in a PL/SQL package called WF_DIRECTORY.

See Also

Standard API for an Oracle Workflow PL/SQL Stored Procedure: page 6 – 2

GetRoleUsers

Syntax `procedure GetRoleUsers`
 `(role in varchar2,`
 `users out UserTable);`

Description Returns a table of users for a given role.

Arguments (input) **role** A valid role name.

GetUserRoles

Syntax `procedure GetUserRoles`
 `(user in varchar2,`
 `roles out RoleTable);`

Description Returns a table of roles that a given user is assigned to.

Arguments (input) **user** A valid username.

GetRoleInfo

Syntax `procedure GetRoleInfo`
 `(Role in varchar2,`
 `Display_Name out varchar2,`
 `Email_Address out varchar2,`
 `Notification_Preference out varchar2,`
 `Language out varchar2,`
 `Territory out varchar2);`

Description Returns the following information about a role:

- Display name
- Email address
- Notification Preference ('QUERY', 'MAILTEXT', 'MAILHTML', 'SUMMARY')
- Language
- Territory

Arguments (input) **role** A valid role name.

IsPerformer

Syntax

```
function IsPerformer
    (user in varchar2,
     role in varchar2);
```

Description Returns true or false to identify whether a user is a performer of a role.

Arguments (input)	user	A valid username.
	role	A valid role name.

CurrentUser

Syntax `function CurrentUser
 return varchar2;`

Description Returns the current Application Object Library username. This function is useful only for the version of Oracle Workflow embedded in Oracle Applications.

UserActive

Syntax `function UserActive
 (username in varchar2)
 return boolean;`

Description Determines if a user is currently active and available to participate in a workflow. Returns TRUE if the user is active, otherwise it returns FALSE.

GetUserName

Syntax `procedure GetUserName`
 `(p_orig_system in varchar2,`
 `p_orig_system_id in varchar2,`
 `p_name out varchar2,`
 `p_display_name out varchar2);`

Description Returns a Workflow display name and username for a user given the system information from the original user and roles repository.

Arguments (input) **p_orig_system** Code that identifies the original repository table.
 p_orig_system_id ID of a row in the original repository table.

GetRoleName

Syntax

```
procedure GetRoleName
    (p_orig_system in varchar2,
    p_orig_system_id in varchar2,
    p_name out varchar2,
    p_display_name out varchar2 );
```

Description Returns a Workflow display name and role name for a role given the system information from the original user and roles repository.

Arguments (input)

p_orig_system	Code that identifies the original repository table.
p_orig_system_id	ID of a row in the original repository table.

Workflow Monitor APIs

Call the following APIs to generate a complete URL to access the various pages of the Workflow Monitor. The APIs are defined in the PL/SQL package called WF_MONITOR.



Attention: The GetURL API from earlier versions of Oracle Workflow is now replaced by the GetEnvelopeURL and GetDiagramURL APIs. The functionality of the previous GetURL API correlates directly with the new GetDiagramURL API. The current version of Oracle Workflow still recognizes the GetURL API, but moving forward, you should only use the two new APIs where appropriate.

GetDiagramURL

Syntax

```
function GetDiagramURL
    (x_agent in varchar2,
    x_item_type in varchar2,
    x_item_key in varchar2,
    x_admin_mode in varchar2 default 'NO')
    return varchar2;
```

Description Can be called by an application to return a URL that allows access to the Workflow Monitor with an attached access key password. The URL displays the diagram for a specific instance of a workflow process in the Workflow Monitor operating in either 'ADMIN' or 'USER' mode.

The URL returned by the function *WF_MONITOR.GetDiagramURL()* looks as follows:

```
<webagent>/wf_monitor.html?x_item_type=
<item_type>&x_item_key=<item_key>&x_admin_mode=<YES or NO>
&x_access_key=<access_key>
```

<webagent> represents the base URL of the Oracle Web Agent used by Oracle Workflow. It looks something like http://<server.com:portID>/<PLSQL_agent_virtual_path>. See: Identifying the Oracle Web Agent used by Oracle Workflow: page 2 - 15.

wf_monitor.html represents the name of the PL/SQL package procedure that generates the Workflow Monitor diagram of the process instance.

The `wf_monitor.html` procedure requires four arguments. `<item_type>` and `<item_key>` represent the internal name of the item type and the item key that uniquely identify an instance of a process. If `<YES or NO>` is YES, the monitor runs in 'ADMIN' mode and if NO, the monitor runs in 'USER' mode. `<access_key>` represents the access key password that determines whether the monitor is run in 'ADMIN' or 'USER' mode.

Arguments (input)	x_agent	<p>The base web agent string defined for Oracle Workflow or Oracle Self-Service Web Applications in Oracle WebServer. The base web agent string should be stored in the WF_RESOURCES table, and looks something like:</p> <pre>http://<server.com:portID>/<PLSQL_agent_path></pre> <p>When calling this function, your application must first retrieve the web agent string from the WF_RESOURCES token WF_WEB_AGENT by calling <code>WF_CORE.TRANSLATE()</code>. See: Identifying the Oracle Web Agent used by Oracle Workflow: page 2 – 15.</p>
	x_item_type	A valid item type.
	x_item_key	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process to report on.
	x_admin_mode	A value of YES or NO. YES directs the function to retrieve the access key password that runs the monitor in 'ADMIN' mode. NO retrieves the access key password that runs the monitor in 'USER' mode.

Example Following is an example of how you can call the `GetDiagramUrl`. This example returns a URL that displays the Workflow Monitor diagram for a process instance identified by the item type WFDEMO and item key 10022, in 'USER' mode:

```
URL := WF_MONITOR.GetDiagramURL
      (WF_CORE.Translate('WF_WEB_AGENT'),
       'WFDEMO',
       '10022',
       'NO');
```

GetEnvelopeURL

Syntax

```
function GetEnvelopeURL
    (x_agent in varchar2,
     x_item_type in varchar2,
     x_item_key in varchar2,
     x_admin_mode in varchar2 default 'NO')
    return varchar2;
```

Description Can be called by an application to return a URL that allows access to the Workflow Monitor Notifications List with an attached access key password. The URL displays the Notifications List for a specific instance of a workflow process in the Workflow Monitor.

The URL returned by the function *WF_MONITOR.GetEnvelopeURL()* looks as follows:

```
<webagent>/wf_monitor.envelope?x_item_type=
<item_type>&x_item_key=<item_key>&x_admin_mode=<YES or NO>
&x_access_key=<access_key>
```

<webagent> represents the base URL of the Oracle Web Agent used by Oracle Workflow. It looks something like `http://<server.com:portID>/<PLSQL_agent_virtual_path>`. See: [Identifying the Oracle Web Agent used by Oracle Workflow: page 2 – 15.](#)

wf_monitor.envelope represents the name of the PL/SQL package procedure that generates the Workflow Monitor Notifications List for the process instance.

The *wf_monitor.envelope* procedure requires four arguments. *<item_type>* and *<item_key>* represent the internal name of the item type and the item key that uniquely identify an instance of a process. If *<YES or NO>* is YES, the monitor runs in 'ADMIN' mode and if NO, the monitor runs in 'USER' mode. *<access_key>* represents the access key password that determines whether the monitor is run in 'ADMIN' or 'USER' mode.

Arguments (input)	x_agent	<p>The base web agent string defined for Oracle Workflow or Oracle Self-Service Web Applications in Oracle WebServer. The base web agent string should be stored in the WF_RESOURCES table, and looks something like:</p> <pre>http://<server.com:portID>/<PLSQL_agent_path></pre> <p>When calling this function, your application must first retrieve the web agent string from the WF_RESOURCES token WF_WEB_AGENT by calling <i>WF_CORE.TRANSLATE()</i>. See: Identifying the Oracle Web Agent used by Oracle Workflow: page 2 – 15.</p>
	x_item_type	A valid item type.
	x_item_key	A string generated from the application object's primary key. The string uniquely identifies the item within an item type. The item type and key together identify the process to report on.
	x_admin_mode	A value of YES or NO. YES directs the function to retrieve the access key password that runs the monitor in 'ADMIN' mode. NO retrieves the access key password that runs the monitor in 'USER' mode.

See Also

TRANSLATE: page 7 – 41

Oracle Workflow Views

Public views are available for accessing workflow data. If you are using the version of Oracle Workflow embedded in Oracle Applications, these views are installed in the APPS account. If you are using the standalone version of Oracle Workflow, these view are installed in the same account as the Oracle Workflow server.

WF_ITEM_ACTIVITY_STATUSES_V

This view contains denormalized information about a workflow process and its activities' statuses. Use this view to create custom queries and reports on the status of a particular item or process. The column descriptions of the view are as follows:

Name	Null?	Type
-----	-----	-----
ROW_ID		ROWID
SOURCE		CHAR(1)
ITEM_TYPE		VARCHAR2(8)
ITEM_TYPE_DISPLAY_NAME		VARCHAR2(80)
ITEM_TYPE_DESCRIPTION		VARCHAR2(240)
ITEM_KEY		VARCHAR2(240)
USER_KEY		VARCHAR2(240)
ITEM_BEGIN_DATE		DATE
ITEM_END_DATE		DATE
ACTIVITY_ID		NUMBER
ACTIVITY_LABEL		VARCHAR2(30)
ACTIVITY_NAME		VARCHAR2(30)
ACTIVITY_DISPLAY_NAME		VARCHAR2(80)
ACTIVITY_DESCRIPTION		VARCHAR2(240)
ACTIVITY_TYPE_CODE		VARCHAR2(8)
ACTIVITY_TYPE_DISPLAY_NAME		VARCHAR2(80)
EXECUTION_TIME		NUMBER
ACTIVITY_BEGIN_DATE		DATE
ACTIVITY_END_DATE		DATE
ACTIVITY_STATUS_CODE		VARCHAR2(8)
ACTIVITY_STATUS_DISPLAY_NAME		VARCHAR2(80)
ACTIVITY_RESULT_CODE		VARCHAR2(30)
ACTIVITY_RESULT_DISPLAY_NAME		VARCHAR2(2000)
ASSIGNED_USER		VARCHAR2(30)
ASSIGNED_USER_DISPLAY_NAME		VARCHAR2(2000)
NOTIFICATION_ID		NUMBER

ERROR_NAME	VARCHAR2 (30)
ERROR_MESSAGE	VARCHAR2 (2000)
ERROR_STACK	VARCHAR2 (2000)

WF_NOTIFICATION_ATTR_RESP_V

This view contains information about the Respond message attributes for a notification group. If you plan to create a custom "voting" activity, use this view to create the function that tallies the responses from the users in the notification group. See: Voting Activity: page 5 – 8.

The column descriptions of the view are as follows:

Name	Null?	Type
GROUP_ID	NOT NULL	NUMBER
RECIPIENT_ROLE	NOT NULL	VARCHAR2(30)
RECIPIENT_ROLE_DISPLAY_NAME		VARCHAR2(2000)
ATTRIBUTE_NAME	NOT NULL	VARCHAR2(30)
ATTRIBUTE_DISPLAY_NAME	NOT NULL	VARCHAR2(80)
ATTRIBUTE_VALUE		VARCHAR2(2000)
ATTRIBUTE_DISPLAY_VALUE		VARCHAR2(2000)
MESSAGE_TYPE	NOT NULL	VARCHAR2(8)
MESSAGE_NAME	NOT NULL	VARCHAR2(30)

WF_RUNNABLE_PROCESSES_V

This view contains a list of all runnable workflow processes in the ACTIVITIES table.

The column descriptions of the view are as follows:

Name	Null?	Type
-----	-----	-----
ITEM_TYPE	NOT NULL	VARCHAR2 (8)
PROCESS_NAME	NOT NULL	VARCHAR2 (30)
DISPLAY_NAME	NOT NULL	VARCHAR2 (80)

WF_ITEMS_V

This view is a select only version of the WF_ITEMS table.

The column descriptions of the view are as follows:

Name	Null?	Type
-----	-----	-----
ITEM_TYPE	NOT NULL	VARCHAR2(8)
ITEM_KEY	NOT NULL	VARCHAR2(240)
ROOT_ACTIVITY	NOT NULL	VARCHAR2(30)
ROOT_ACTIVITY_VERSION	NOT NULL	NUMBER
OWNER_ROLE		VARCHAR2(30)
PARENT_ITEM_TYPE		VARCHAR2(8)
PARENT_ITEM_KEY		VARCHAR2(240)
PARENT_CONTEXT		VARCHAR2(2000)
BEGIN_DATE	NOT NULL	DATE
END_DATE		DATE

Overview of Notification APIs

Oracle Workflow communicates with users by sending notifications. Notifications may request users to take some type of action and/or provide users with information. You define the notification activity and the message that the notification activity sends in the Workflow Builder. The messages may have optional attributes that can specify additional resources and request responses.

Users can query their notifications online using either the Notification Viewer in Oracle Applications or the Notifications web page in an HTML browser. Alternatively, a user can receive notifications via E-mail. E-mail notifications can include HTML pages as optional attachments. The Notification System delivers the messages and processes the incoming responses.

Notification Model

The notification activities in a workflow process use a list of design-time messages and message attributes. In addition, there are a number of runtime named values called item type attributes from which the message attributes draw their values.

The Workflow Engine moves through the process, evaluating each activity in turn. Once a notification activity is encountered, the engine makes a call to the Notification System *Send()* or *SendGroup()* API to handle the notification.

The *Send()* or *SendGroup()* API looks up the message attributes for the message associated with the current activity. If a particular attribute is of source SEND, its value is looked up from the item type attribute associated with that message attribute as defined in the Workflow Builder. If no item type attribute can be found, the default value of the message attribute is used, if available. If the attribute is of source RESPOND, the procedure checks to see if it has a default value assigned to it.

The message Subject and Body may include message attributes of source SEND, which the *Send()* or *SendGroup()* API replaces with the attributes' current values when creating the notification. If a performer's notification preference is 'MAIL' or 'MAILHTML', the *Send()* or *SendGroup()* API also creates a notification E-mail message. Otherwise, a performer can always view notifications from a Web browser.

If the message includes message attributes of source RESPOND, which are used to prompt the performer for a response, the *Send()* or *SendGroup()* API creates a response section in the notification.

After a user responds, HTML browser or parsing mail agent analyzes the response and updates the respond value by calling the notification *Respond()* API to pass the respond values to the Workflow Engine. The engine then updates the corresponding item type attributes. Also, if one of the RESPOND message attributes is named 'RESULT', its value becomes the actual result of the notification activity.

Respond() then calls the Workflow Engine *CompleteActivity()* API to inform the engine that the notification activity is complete and it can now transition to the next qualified activity, as long as a callback function is defined in *Send()*.

Note: If a notification activity sends a message that is for the performer's information only, where there are no Respond message attributes associated with it, the notification activity gets marked as complete as soon as the Notification System delivers the message.

Notification APIs

The following APIs can be called by a notification agent to manage notifications for a notification activity. The APIs are stored in the PL./SQL package called WF_NOTIFICATION.

Send

Syntax function SEND

```
(role in varchar2,  
msg_type in varchar2,  
msg_name in varchar2,  
due_date in date default null,  
callback in varchar2 default null,  
context in varchar2 default null,  
send_comment in varchar2 default null  
priority in number default null)  
return number;
```

Description This function sends the specified message to a role, returning a notification ID if successful. The notification ID must be used in all future references to the notification.

If your message has message attributes of source SEND or RESPOND, the procedure looks up the values of the attributes from the message attribute table or it can use an optionally supplied callback interface function to get the value from the item type attributes table. A callback function can also be used when a notification is responded to. The syntax of the callback function should follow the example described below.

Arguments (input)	role	The role name assigned as the performer of the notification activity.
	msg_type	The item type associated with the message.
	msg_name	The message internal name.
	due_date	The date that a response is required. This optional due date is only for the recipient's information; it has no effect on processing.
	callback	The callback function name used for communication of SEND and RESPOND source message attributes.

context	Context information passed to the callback function.
send_comment	A comment presented with the message.
priority	The priority of the message, as derived from the #PRIORITY notification activity attribute. If #PRIORITY does not exist or if the value is null, the Workflow Engine uses the default priority of the message.

The callback function must have the following specifications:

```
procedure <name in callback argument>
(command in varchar2,
context in varchar2,
attr_name in varchar2,
attr_type in varchar2,
text_value in out varchar2,
number_value in out number,
date_value in out date);
```

Arguments (input)

command	Specify GET, SET, COMPLETE or ERROR. Use GET to get the value of an attribute, SET to set the value of an attribute, COMPLETE to indicate that the response is complete and ERROR to set the associated notification activity to a status of 'ERROR'.
context	The context passed to <i>SEND()</i> . The format is <i><itemtype>:<itemkey>:<activityid></i> .
attr_name	An attribute name.
attr_type	An attribute type.
text_value	Value of a text attribute.
number_value	Value of a number attribute.
date_value	Value of a date attribute.

When a notification is sent, the system calls the specified callback function once for each SEND attribute (to get the attribute value).

Example 1 For each SEND attribute, call:

```
your_callback('GET', context, 'BUGNO')
```

Example 2 When the user responds to the notification, the callback is called again, once for each RESPOND attribute.

```
your_callback('SET', context, 'STATUS');
```

Example 3 Then finally use the 'COMPLETE' command to indicate the response is complete.

```
your_callback('COMPLETE', context);
```

SendGroup

Syntax

```
function SendGroup
(role in varchar2,
 msg_type in varchar2,
 msg_name in varchar2,
 due_date in date default null,
 callback in varchar2 default null,
 context in varchar2 default null,
 send_comment in varchar2 default null
 priority in number default null)
return number;
```

Description This function sends a separate notification to all the users assigned to a specific role and returns a number called a notification group ID, if successful. The notification group ID identifies that group of users and the notification they each received.

If your message has message attributes of source SEND or RESPOND, the procedure looks up the values of the attributes from the message attribute table or it can use an optionally supplied callback interface function to get the value from the item type attributes table. A callback function can also be used when a notification is responded to. See: Send: page 7 – 69.

Generally, this function is called only if a notification activity has 'Expanded Roles' checked in its properties page. If Expanded Roles is not checked, then the *Send()* function is called instead. See: Voting Activity: page 5 – 8.

Arguments (input)	role	The role name assigned as the performer of the notification activity.
	msg_type	The item type associated with the message.
	msg_name	The message internal name.
	due_date	The date that a response is required. This optional due date is only for the recipient's information; it has no effect on processing.
	callback	The callback function name used for communication of SEND and RESPOND source message attributes.
	context	Context information passed to the callback function.

send_comment	A comment presented with the message.
priority	The priority of the message, as derived from the #PRIORITY notification activity attribute. If #PRIORITY does not exist or if the value is null, the Workflow Engine uses the default priority of the message.

Forward

Syntax `procedure FORWARD`
`(nid in number,`
`new_role in varchar2,`
`forward_comment in varchar2 default null);`

Description This procedure reassigns a notification to a new role. A comment can be supplied to explain why the forward is taking place. The notification will then be delivered to the new role. Existing notification attributes (including due date) are not refreshed or otherwise changed. The Reassign feature in the Notification System calls this procedure.

Arguments (input)

nid	The notification id.
new_role	The role name of the person the note is reassigned to.
forward_comment	An optional forwarding comment.

Cancel

Syntax `procedure CANCEL
(nid in number,
cancel_comment in varchar2 default null);`

Description This procedure may be invoked by the sender or administrator to cancel a notification. The notification status is then changed to 'CANCELED' but the row is not removed from the WF_NOTIFICATIONS table until a purge operation is performed.

If the notification was delivered via e-mail and expects a response, a 'Canceled' e-mail is sent to the original recipient as a warning that the notification is no longer valid.

Arguments (input)

nid	The notification id.
cancel_comment	An optional comment on the cancellation.

CancelGroup

Syntax `procedure CancelGroup`
`(gid in number,`
`cancel_comment in varchar2 default null);`

Description This procedure may be invoked by the sender or administrator to cancel the individual copies of a specific notification sent to all users in a notification group. The notifications are identified by the notification group ID (gid). The notification status is then changed to 'CANCELED' but the rows are not removed from the WF_NOTIFICATIONS table until a purge operation is performed.

If the notification was delivered via e-mail and expects a response, a 'Canceled' e-mail is sent to the original recipient as a warning that the notification is no longer valid.

Generally, this function is called only if a notification activity has 'Expanded Roles' checked in its properties page. If Expanded Roles is not checked, then the *Cancel()* function is called instead. See: Voting Activity: page 5 – 8.

Arguments (input) **gid** The notification group id.
cancel_comment An optional comment on the cancellation.

Respond

Syntax `procedure RESPOND`
`(nid in number,`
`respond_comment in varchar2 default null,`
`responder in varchar2 default null);`

Description This procedure may be invoked by the notification agent (Notification Viewer, Notification Web page, or E-mail agent) when the performer completes the response to the notification. The procedure marks the notification as 'CLOSED' and communicates RESPOND attributes back to the database via the callback function (if supplied). If one of the RESPOND attributes is named 'RESULT', that RESPOND attribute's value is instead used as the result of the associated notification activity.

This procedure also accepts the name of the individual that actually responded to the notification. This may be useful to know especially if the notification is assigned to a multi-user role. The information is stored in the RESPONDER column of the WF_NOTIFICATIONS table. The value stored in this column depends on how the user responds to the notification.

Response Mechanism	Value Stored
Web	Web login username
Oracle Applications Notifications Viewer Form	Oracle Applications login username
E-Mail	E-mail username as displayed in the mail response.

Arguments (input)

nid	The notification id
comment	An optional comment on the response
responder	The user who responded to the notification.

Responder

Syntax `function RESPONDER`
 `(nid in number)`
 `returns varchar2;`

Description This function returns the responder of a closed notification.

If the notification was closed using the Notification Viewer form or Web Notification interface the value returned will be a valid role defined in the view WF_ROLES. If the Notification was closed using the E-mail interface then the value returned will be an E-mail address.

Arguments (input) **nid** The notification id

VoteCount

Syntax `procedure VoteCount`
 `(gid in number,`
 `resultCode in varchar2,`
 `resultCount out number,`
 `percentOfTotalPop out number,`
 `percentOfVotes out number);`

Description Counts the number of responses for a specified result code.

Use this procedure only if you are writing your own custom Voting activity. See: Voting Activity: page 5 – 8.

Arguments (input)

gid	The notification group id.
resultCode	Result code to be tallied.

OpenNotificationsExist

Syntax `function OpenNotificationsExist`
 `(gid in number)`
 `return boolean;`

Description This function returns a value of 'TRUE' if any notification associated with the specified notification group ID is 'OPEN', otherwise it returns a value of 'FALSE'.

Use this procedure only if you are writing your own custom Voting activity. See: Voting Activity: page 5 – 8.

Arguments (input) **gid** The notification group id.

AddAttr

Syntax `procedure AddAttr`
 `(nid in number,`
 `aname in varchar2);`

Description Adds a new runtime notification attribute. You should perform validation and insure consistency in the use of the attribute, as it is completely unvalidated by Oracle Workflow.

Arguments (input)	nid	The notification id.
	aname	The attribute name.
	avalue	The attribute value.

SetAttribute

Syntax

```
procedure SetAttrText
    (nid in number,
     aname in varchar2,
     avalue in varchar2);

procedure SetAttrNumber
    (nid in number,
     aname in varchar2,
     avalue in number);

procedure SetAttrDate
    (nid in number,
     aname in varchar2,
     avalue in date);
```

Description Used at both send and respond time to set the value of notification attributes. The notification agent (sender) may set the value of SEND attributes. The performer (responder) may set the value of RESPOND attributes.

Arguments (input)	nid	The notification id.
	aname	The attribute name.
	avalue	The attribute value.

GetAttrInfo

Syntax `procedure GetAttrInfo`
`(nid in number,`
`aname in varchar2,`
`atype out varchar2,`
`subtype out varchar2,`
`format out varchar2);`

Description Returns information about a notification attribute, such as its type, subtype, and format, if any is specified. The subtype is always SEND or RESPOND to indicate the attribute's source.

Arguments (input)	nid	The notification id.
	aname	The attribute name.

GetInfo

Syntax procedure GetInfo

 (nid in number,
 role out varchar2,
 message_type out varchar2,
 message_name out varchar2,
 priority out number,
 due_date out date,
 status out varchar2);

Description Returns various attributes for the specified notification.

Arguments (input)	nid	The notification id.
	role	Role name of the performer assigned to the notification activity.
	message_type	The item type associated with the message.
	message_name	The message name.
	priority	The notification priority.
	due_date	The due date of the notification.
	status	The current status of the notification: OPEN, CLOSED, CANCELED or ERROR.

GetText

Syntax

```
function GetText
    (some_text in varchar2,
     nid in number)
    return varchar2;
```

Description Substitutes tokens in an arbitrary text string using token values from a particular notification. If an error is detected, this function returns `some_text` unsubstituted rather than raise exceptions.

Arguments (input)	some_text	Text to be substituted.
	nid	Notification ID of notification to use for token values.

GetAttribute

Syntax

```
function GetAttrText
    (nid in number,
     aname in varchar2)
    return varchar2;

function GetAttrNumber
    (nid in number,
     aname in varchar2)
    return number;

function GetAttrDate
    (nid in number,
     aname in varchar2)
    return date;
```

Description Returns the value of the specified message attribute.

Arguments (input)	nid	The notification id.
	aname	The message attribute name.

GetSubject

Syntax `function GetSubject
 (nid in number)
 return varchar2`

Description Returns the subject line for the notification message. Any message attribute in the subject is token substituted with the value of the corresponding message attribute.

Argument (input) **nid** The notification id

GetBody

Syntax

```
function GetBody  
(nid in number)  
return varchar2;
```

Description Returns the message body for the notification. Any message attribute in the body is token substituted with the value of the corresponding message attribute. This text is *not* formatted; it should be wordwrapped as appropriate for the output device. Body text may contain tabs (which indicate indentation) and newlines (which indication paragraph termination).

Argument (input) **nid** The notification id

AccessCheck

Syntax

```
function AccessCheck  
(access_str in varchar2)  
return varchar2;
```

Description Returns a username if the notification access string is valid and the notification is open, otherwise it returns null. The access string is used to verify the authenticity of both text and HTML versions of E-mail notifications.

Argument (input) **access_str** The access string, in the format:
nid/nkey
where *nid* is the notification ID and *nkey* is the notification key.

WorkCount

Syntax `function WorkCount`
 `(username in varchar2)`
 `return number;`

Description Returns the number of open notifications assigned to a role.

Argument (input) **username** The internal name of a role.

CHAPTER

8

Viewing Notifications and Processing Responses

This chapter discusses the different ways people involved in a workflow process can view and respond to workflow notifications. This chapter also describes how you can define rules to have Oracle Workflow automatically handle your notifications.

Overview of Notification Handling

Oracle Workflow sends a notification to a role when the Workflow Engine executes a notification activity in a workflow process. The notification activity may designate the role as being responsible for performing some human action or may simply relay process-related information to the role. To successfully deliver a notification to a role, the role must be defined in the Oracle Workflow directory service.

As a member of a role, you can view a notification using any one of four interfaces depending on your role's notification preference setting in the Oracle Workflow directory service. You can receive an E-mail for each individual notification, receive a single E-mail summarizing all your notifications, query the Workflow Notifications Web page or query the Workflow Notification Viewer form (for Oracle Applications users only) for your notifications. See: *Setting Up an Oracle Workflow Directory Service*: page 2 – 7.

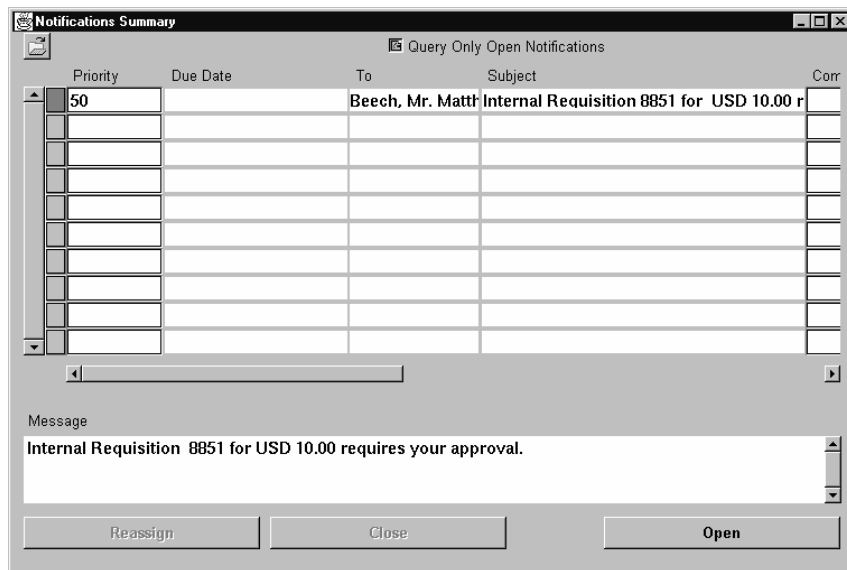
Each notification message can include context-sensitive information about the process and directions on how to respond to the notification, if a response is required. The message can also include pointers to Web URLs and references to Oracle Applications forms that allow the user to get additional information related to the notification.

As a notification recipient, there may be occasions when you will not be able to view or respond to your notifications in a timely manner. Rather than create a bottleneck in a workflow process, you can take advantage of the Automatic Notification Handler to define rules that direct Oracle Workflow to automatically handle the notifications for you.

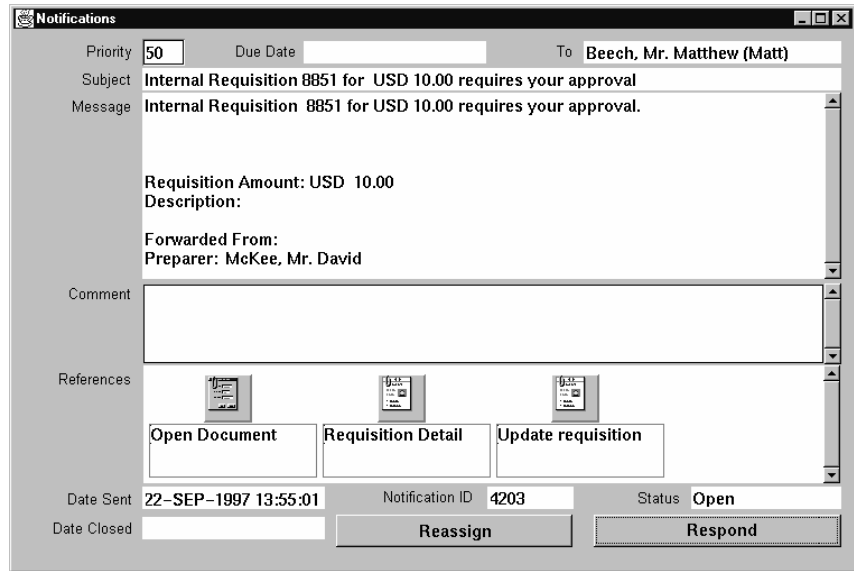
Reviewing Notifications in the Notification Viewer (for Oracle Applications Users Only)

The Notification Viewer is an Oracle Applications form function. To display the Notification Viewer (or "Personal Inbox"), an Oracle Applications System Administrator or Oracle Applications developer must add this form function to the Navigate menu of a user's responsibility or call this form function from another Oracle Applications form. The Notification Viewer developer form name is FNDWFNOT and its function name is FND_FNDWFNOT. See: *Overview of Function Security, Oracle Applications System Administrator's Guide* and *Overview of Menus and Function Security, Oracle Applications Developer's Guide*.

The Notifications Summary window lists all the notifications sent to you. By default, the window queries only open notifications, but you can choose to show all of your notifications.



The fields and prompts on this folder window can be customized. You can drill down on the current record indicator to view details of an individual notification. The body of the message appears in the overflow region at the bottom of the screen. See: Customizing the Presentation of Data in a Folder, *Oracle Applications User's Guide*.



When you select a notification record in the Notifications Summary window, the Notifications window appears, listing the details of that notification. From the Notifications window you can:

- Reassign the notification to another user.
- Respond to the notification or close the notification if it does not require a response.
- Drill down to another Oracle Applications window associated with the notification if icons exist in the References region.

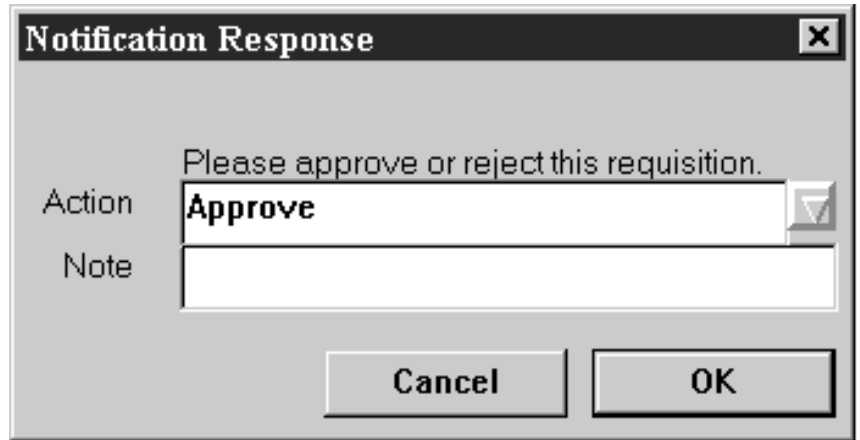
If the notification was reassigned or forwarded from another user, the Comment field may include comments entered by that user.

► **To Reassign a Notification to Another User**

1. Navigate to the Notifications Summary window, also known as your "Personal Inbox". Select the notification to reassign.
2. In the Comment field, enter any comments you wish to pass along with the notification. Note that these comments are associated only with the notification and are not passed on as information to other activities in the workflow process.
3. Choose Reassign in either the Notifications Summary or Notifications window.

4. Choose the role name to whom you want to reassign the notification.
5. Choose OK.

► **To Respond to a Notification**



The image shows a dialog box titled "Notification Response". At the top, there is a header bar with the title and a close button (X). Below the header, there is a text box containing the prompt "Please approve or reject this requisition." Underneath this prompt is a dropdown menu with the word "Approve" selected. To the left of the dropdown menu are the labels "Action" and "Note". Below the dropdown menu is an empty text box for entering a note. At the bottom of the dialog box, there are two buttons: "Cancel" and "OK".

1. Navigate to the Notifications Summary window. Select the notification you want to respond to. The Notifications window appears.
2. Choose Respond to open the Notification Response window if the notification requires a response.

If the notification does not require a response, choose Close to close the notification, so that it does not appear when you query for open or new notifications in the future.

If the notification requires a response, the Notification Response window may require you to supply one or more response values in the fields provided or it may require you to take action in a form.
3. To respond to a notification, enter the information as indicated in the window and choose OK, or complete the action required in the designated form and save your changes.

Once you respond to or close an open notification, you can query for the closed notification again by unchecking the Query Only Open Notifications check box before querying the Notifications Summary window.

You can also choose the View Response button to display the response provided for that notification.

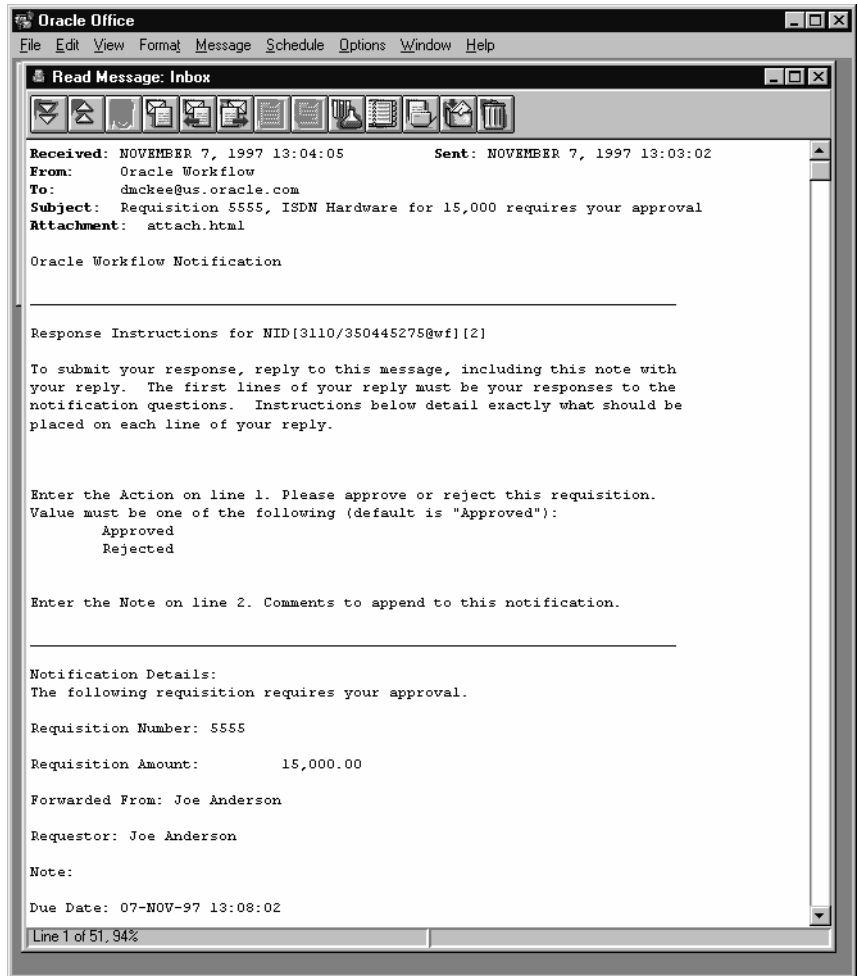
► **To Drill Down to a Form or URL Associated With a Notification:**

1. Navigate to the Notifications Summary window.
2. Double-click on the current record indicator for a notification. The Notifications window appears.
3. In the References region, choose the icon for the reference form or reference URL to which you wish to navigate.
4. When you finish, close the form to return to the Notifications window.

Reviewing Notifications via Electronic Mail

You can have your workflow notifications delivered to you as E-mail messages if your notification preference is set to 'MAILTEXT' or 'MAILHTML' in the Oracle Workflow directory service views. If your E-mail tool supports attachments, an E-mail notification may include an HTML attachment with the content and response fields of the notification formatted in a Web page. The user may respond either by replying to the E-mail as specified by the directions included in the E-mail, or by activating the HTML attachment and filling in the requested information in the resulting Web page.

You can receive E-mail notifications using any mail application that is MAPI-compliant running on Windows NT or that Oracle Office/InterOffice or UNIX Sendmail can provide a gateway to. The following example shows a notification received through Oracle Office.



The E-mail notification is based on a standard template defined in Oracle Workflow Builder. It describes the syntax the reply should follow, and lists the information needed to confirm the notification. The message also identifies any custom site information, specifies the due date of the message, and details any information necessary to process the response. See: *Modifying Your Message Templates*: page 2 – 36.

► **To Respond to a Notification by E-mail**

1. Use the appropriate command in your mail application to reply to the E-mail notification, including the original message in the reply. This ensures that the Notification ID (NID) and the notification

access key is within the text of your reply. The Oracle Workflow Notification Mailer can process your response properly only if you include the correct NID and access key combination with your response.

Note: The notification access key is a distinct random key that the Notification System generates for each NID. The access key serves as a password and prevents users from responding to the notification unless the notification was actually mailed to them since they would not be able to guess the valid NID and access key combination.

2. Follow the syntax instructions in the mail note carefully when formatting your reply. The response values must be within the first lines of your reply, where each line represents a separate response value.

If a response value requires more than one line, then the entire response value must be enclosed in double quotes (" ") and everything enclosed in the double quotes is counted as one line.

The Notification System interprets your response values literally, so a value in uppercase is interpreted differently from the same value in lowercase.

If a response prompt provides a default response value, you can accept the default value by leaving the appropriate response line blank.



Warning: Turn off automatic signatures when you reply to notifications, as they may be interpreted incorrectly as response values. For example, suppose a notification expects four response values to be returned and you specify three response values in the first three lines and then leave the fourth line blank to accept the default value. If you include an automatic signature in the response, the Notification Mailer may incorrectly interpret your signature as the fourth response value.

3. If you send an invalid response, the Notification System sends you an "invalid response" message. If you respond to a notification that has been canceled, you get a message informing you that the notification was canceled. Similarly, if you respond to a notification that was already previously responded to, you get a message informing you that the notification is closed.

Example Following is a set of response instructions and examples of three possible responses.

Response Instructions
Enter the Action on line 1. Do you approve? Value must be one of the following (default is "Reject"):
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. Default is 1500.

Table 8 - 1 (Page 1 of 1)

Valid Response A - Approve
Approve
Let me know if this item meets expectations.
01-JAN-1998
1000.00

Table 8 - 2 (Page 1 of 1)

Valid Response B - Reject
Too expensive.

Table 8 - 3 (Page 1 of 1)

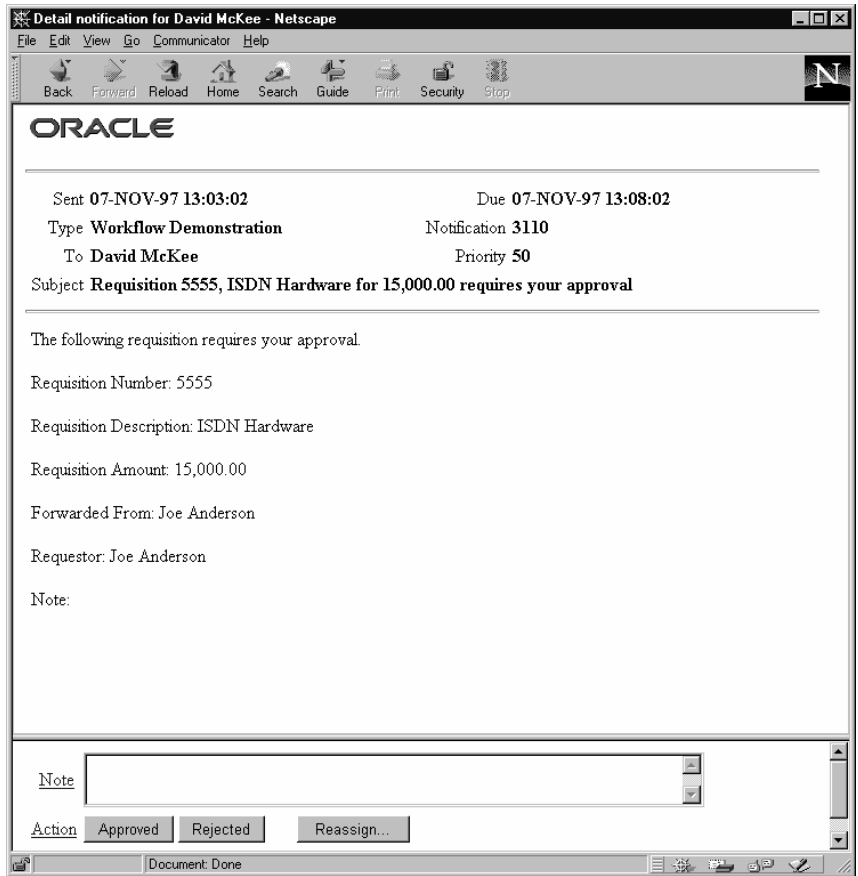
Valid Response C – Reject
"This item is too expensive. Please find a replacement that is of lower cost, or else include additional justification for why this item should be approved."
01-JAN-1998
1000.00

Table 8 – 4 (Page 1 of 1)

► **To Respond to a Notification Using the HTML Attachment**

1. Open the attached document using a Web browser that supports JavaScript and Frames.

Note that when you first open the attached page, it automatically attempts to establish a web session with Oracle WebServer. In doing so, it authenticates your access and verifies that the notification is still open and provides a message if it is otherwise.



2. The attached Web page contains two frames. The top frame contains detailed information about the notification, and the bottom frame contains the response section of the notification. The detail frame may contain links to additional reference URLs that provide information related to the current notification. You can navigate to a reference URL by clicking on a link to open another Web browser window.
3. Supply the information requested in the response frame to complete and submit your response.

► **To Reassign a Notification to Another User:**

- Use the "Forward" feature in your mail application to forward or reassign an E-mail notification to another user. Do not use the "Reassign" button on the HTML attachment.

Viewing Notifications from a Web Browser

You can use any Web browser that supports JavaScript and Frames to view and respond to your notifications.

► To View Notifications from a Web Browser

1. If you are using Oracle Self-Service Web Applications, log on using the Oracle Self-Service Web Applications login page and choose the appropriate link to display the Notifications Worklist page. Skip to Step 10.
2. If you are not using Oracle Self-Service Web Applications, you can access your worklist directly by entering one of the following URLs.

To display your complete worklist, enter:

```
<webagent>/wfa_html.worklist[?orderkey=<orderkey>
&status=<status>&user=<user>]
```

To go to the Find Notifications web page where you can presort and display a subset of your worklist, enter:

```
<webagent>/wfa_html.find
```

The portion of the Worklist URL in square brackets [] represents optional arguments that you can pass (by omitting the square brackets).

Replace the bracketed italicized text in these URLs as follows:

- *<webagent>* represents the base URL of the Oracle Web Agent used by Oracle Workflow. It looks something like `http://<server.com:portID>/<PLSQL_agent_virtual_path>`. See: *Identifying the Oracle Web Agent used by Oracle Workflow: page 2 – 15*.
- *<orderkey>* represents the key with which to order the list of notifications. Valid values include PRIORITY, MESSAGE_TYPE, SUBJECT, BEGIN_DATE, DUE_DATE, END_DATE, and STATUS. If you leave *<orderkey>* null, then the default value is PRIORITY.
- *<status>* represents the status of the notifications you wish to display. Valid values include OPEN, CLOSED, CANCELED, and ERROR. If you leave *<status>* null, the URL will display notifications of any status.
- *<user>* represents the internal name of the role to query notifications for. You can only include this argument if you are

logged in to the current web session as a role with workflow administrator privileges. If your role does not have administrator privileges or if you leave `<user>` blank, the URL displays notifications for your current role. See: Identifying the Oracle Workflow Administrator Role: page 2 – 18.

Note: You can also access the Notifications Worklist and Find Notifications web pages from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 8 – 27.

3. If you are accessing either of these URLs for the first time in your web browser session, Oracle WebServer prompts you for a valid username and password to log on, as these URLs are protected. See: Secure the Workflow Database Connection Descriptor (DCD): page 2 – 21.



Username and Password Required [X]

Enter username for Workflow at
wfdemo.us.support.com:8889:

User Name:

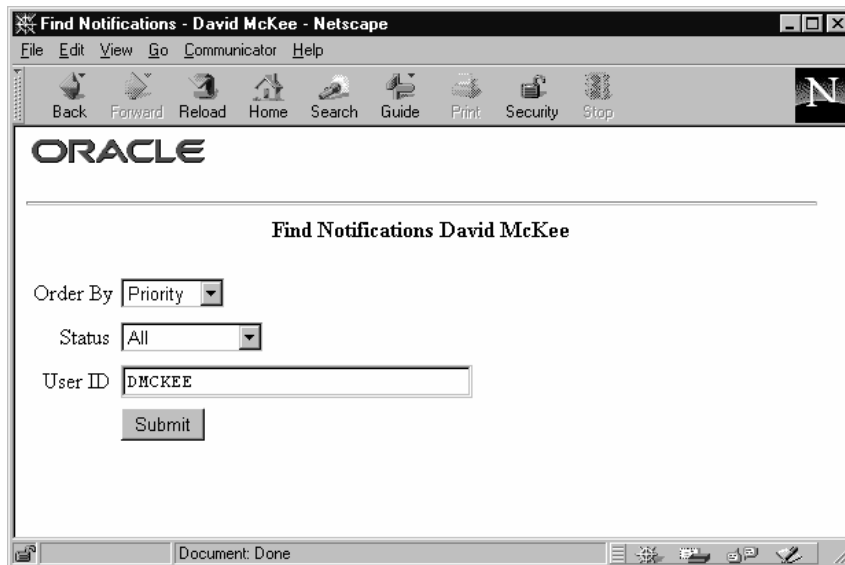
Password:

OK Cancel

4. Enter your username and password.
5. Choose OK. If you have made an error, you can clear the values and start over.

If you used the `wfa_html.worklist` URL skip to the Notification Worklist section: page 8 – 15.

Find Notifications



6. A Find Notifications window appears. Use the Order By field to specify how you wish to order the notifications in the Worklist window. You can choose:
 - Closed – Orders by date for closed processes followed by remaining processes.
 - Due – Orders by due date.
 - Priority – Orders by priority.
 - Sent – Orders by date sent.
 - Status – Lists Open process first followed by closed processes and finally cancelled processes.
 - Subject – Ordered alphabetically using the Subject column.
 - Type – Ordered alphabetically using the Type column.
7. You can use the Status field to restrict the notifications that appear in the Worklist to those that have a specific status. You can choose:
 - All – Displays all processes.
 - Cancelled – Displays only Cancelled processes.
 - Closed – Displays only Closed processes.
 - Invalid Reply – Displays only processes with Invalid Replies.

- Open – Displays only Open processes.
8. If you log on as a role that has workflow administrator privileges, you get an additional field called User ID in the Find Notifications window. You can enter a username in this field to access another user's Worklist. See: Identifying the Oracle Workflow Administration Role: page 2 – 18
 9. Choose the Submit button to open the Worklist window.

Notification Worklist

[Show all notifications](#)

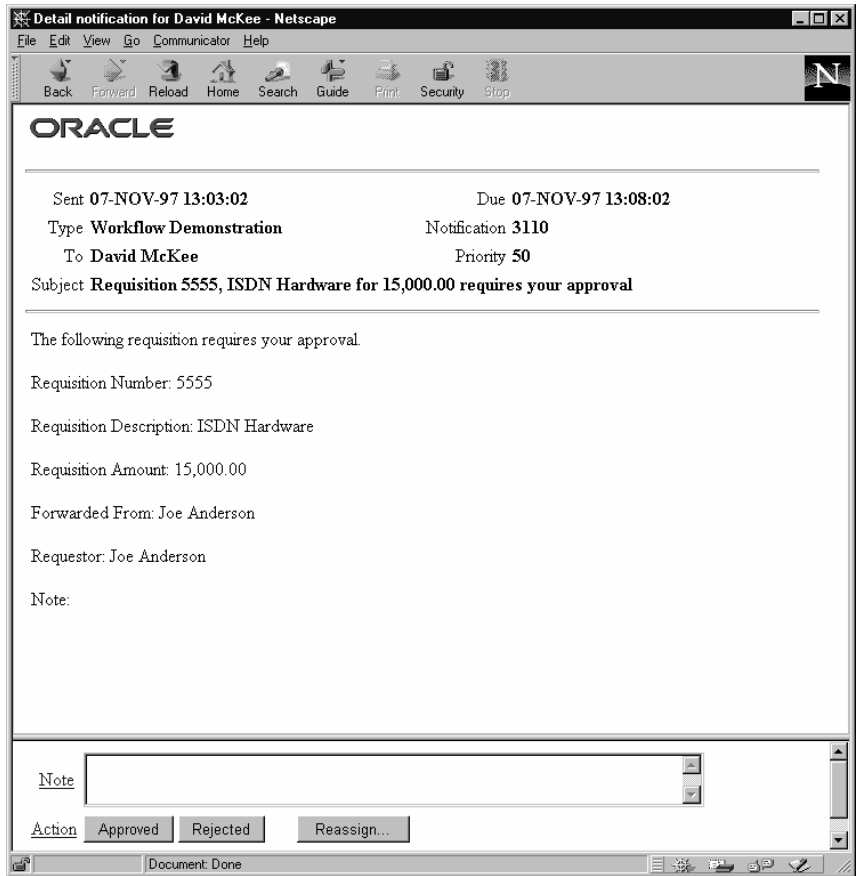
Priority	Type	Subject	Sent	Due
50	Workflow Demonstration	Requisition 5555, ISDN Hardware for 15,000.00 requires your approval	07-NOV-97 13:03:02	07-NOV-97 13:08:02
50	Workflow Demonstration	REMINDER: Requisition 5274, color monitor for 1,001.00 requires your approval	03-NOV-97 14:38:29	03-NOV-97 15:38:29
50	Workflow Demonstration	Requisition 19898, WF-101 Course Enrollment has been forwarded to Matthew Beech for approval	03-NOV-97 23:30:24	
50	Workflow Demonstration	Requisition 19899, HW-200 has been forwarded to Matthew Beech for approval	03-NOV-97 23:32:29	
50	Workflow Demonstration	Requisition 19900, SW-200 has been forwarded to Matthew Beech for approval	03-NOV-97 23:32:54	
50	Workflow Demonstration	Requisition 19898, WF-101 Course Enrollment has been forwarded to John Rush for approval	03-NOV-97 23:38:47	
50	Workflow Demonstration	Requisition 19899, HW-200 has been rejected	03-NOV-97 23:39:40	
50	Workflow Demonstration	Requisition 19900, SW-200 has been forwarded to John Rush for approval	03-NOV-97 23:39:59	
50	Workflow Demonstration	Requisition 19898, WF-101 Course Enrollment has been forwarded to John	03-NOV-97 23:41:44	

10. The Notifications Summary page appears. You can click on ["Show all notifications"](#) / ["Show open notifications only"](#) on the top of the page to toggle between displaying all your notifications or just those that have a status of 'Open', respectively.

The following information is listed depending on whether you are displaying all notifications or just open notifications:

- Priority—a value that represents the importance of the message.
 - Type—the item type that the workflow process and notification is associated with.
 - Subject—a description of the notification.
 - Sent—date and time when the notification was delivered.
 - Due—date and time by which the notification should be completed.
 - Closed—date and time when notification is completed and closed.
 - Status—value that indicates whether the notification is open or closed.
11. Click on any of the column headings to sort your notifications by that column in ascending order.
 12. Select a notification by clicking on a notification subject.

Notifications that you have previously viewed in the current session appear in different color text in the Subject field.



13. A new page appears that displays the details of the notification in an upper frame, and the response section of the notification in a lower frame. You can scroll through or resize either of these frames.
14. The Response section may look as follows:
 - If a notification requires a response, but none of the responses affect the result of the notification activity, the response prompts all appear as fields and/or poplists. When you are done entering your response values, submit your response by choosing the Submit button.
 - If a notification requires a response, and one of the responses becomes the result of the notification activity, then that determining response will appear as a set of buttons to choose from as shown in the figure above. The determining response also appears as the last prompt in the Response section. All

other response prompts, if any, appear as fields or poplists above that prompt. When you choose a button for that last response prompt, you also submit your response for the notification.

- If a notification does not require a response, choose Close in the Response section to close the notification so that it does not appear in your notification summary list the next time you query for open notifications.

Note: You can click on any response prompt to link to a page that displays more information about the response attribute.

15. If you revisit a notification that you just Responded to or Closed in the same session, the Response section shows the past response if one was required.

► **To Reassign a Notification to Another User**

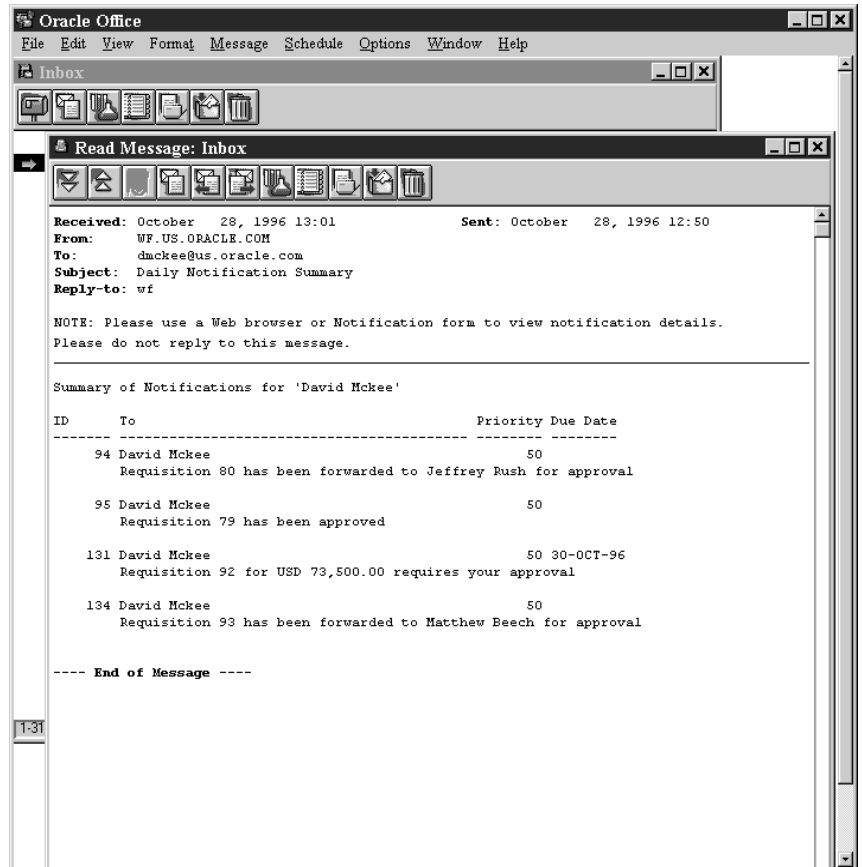
1. Select the notification you want to reassign in your notification summary page.
2. In the Response section of the notification detail page, choose Reassign.
3. In the Notification Reassignment region, enter the username you want to reassign the notification to and enter any comments you want to pass along to that person. Choose Reassign.
4. A confirmation message appears below the notification detail section.
5. If you revisit a notification that you just reassigned to another user, a "Request Failed" message appears to indicate that the notification no longer exists in your list.

If you reload the notification summary page, the reassigned notification no longer appears in your summary list.

Reviewing a Summary of Your Notifications via Electronic Mail

You can have a summary of your workflow notifications delivered to you as a single E-mail message if your notification preference is set to 'SUMMARY' in the Oracle Workflow directory service. The frequency that you receive notification summaries depends on how frequently your Notification Mailer for notification summaries is scheduled to run. See: Starting the Notification Mailer: 2 – 26.

You can receive your E-mail notification summary using any mail application that is MAPI-compliant or that Oracle Office/InterOffice or UNIX Sendmail can provide a gateway to. The following example shows a notification summary received through Oracle Office.



The E-mail notification summary is based on a standard template defined in Oracle Workflow Builder. The summary identifies the recipient, notification ID, subject, priority and due date of each notification. See: *Modifying Your Message Templates*: page 2 – 36.

It also indicates that if you wish to view the details of the notification or respond or close the notification, you should use the Notification Web page or the Notification Viewer form.

Defining Rules for Automatic Notification Handling

Use the Oracle Workflow Automatic Notification Handler to automatically forward your notifications to another role or respond to incoming notifications with a predefined response when you are not available to manage your notifications directly, such as when you are on vacation.

The Notification Routing Rules web page lets you define the rules for the automatic notification handling. Each rule is specific to a role and can apply to any or all messages of a specific item type and/or message name. A rule can perform any one of the following actions: Forward, Respond, or No Action.

Each time the Notification System sends or reassigns a notification to a role, the Automatic Notification Handler tests the notification against that role's list of rules for the most specific match based on the criteria in the order listed below:

ROLE = *<role>* and:

1. MESSAGE_TYPE = *<type>* and MESSAGE_NAME = *<name>*
2. MESSAGE_TYPE = *<type>* and MESSAGE_NAME is null
3. MESSAGE_TYPE is null and MESSAGE_NAME is null

As soon as it finds a match, the Automatic Notification Handler applies the rule and discontinues any further rule matching.

If a rule has an action of Forward, the Automatic Notification Handler performs rule matching again against the new recipient role's list of rules. The Automatic Notification Handler maintains a count of the number of times it forwards a notification to detect perpetual forwarding cycles. If a notification is automatically forwarded more than ten times, the Automatic Notification Handler assumes a forwarding cycle has occurred and ceases executing any further forwarding rules, marking the notification as being in error.

► To Define a Rule for Automatic Notification Routing

1. Use a web browser to connect to one of two URLs.

To display the list of routing rules for your current role, enter:

```
<webagent>/wf_route.list[?user=<rolename>]
```

This URL can include an optional argument, as denoted by the square brackets []. You should omit the square brackets to pass the optional argument.

Replace the bracketed italicized text in the above URL as follows:

- *<webagent>* represents the base URL of the Oracle Web Agent used by Oracle Workflow. It looks something like `http://<server.com:portID>/<PLSQL_agent_virtual_path>`. See: Identifying the Oracle Web Agent used by Oracle Workflow: page 2 – 15.
- *<rolename>* represents an internal role name that you want to query routing rules for. Note, however, that you can query for roles other than your current role only if your current role has workflow administrator privileges. See: Identifying the Oracle Workflow Administration Role: page 2 – 18.

To display a web page that lets you find the routing rules for a specified role, enter:

```
<webagent>/wf_route.find
```

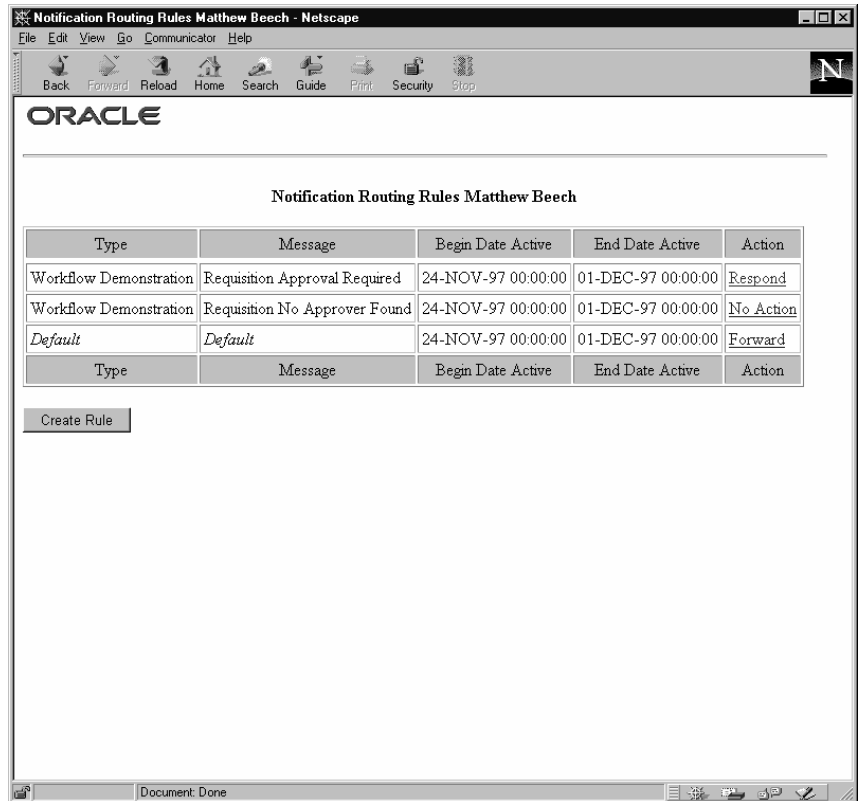


Note: You can also access the Find Notification Routing Rules web pages from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 8 – 27.

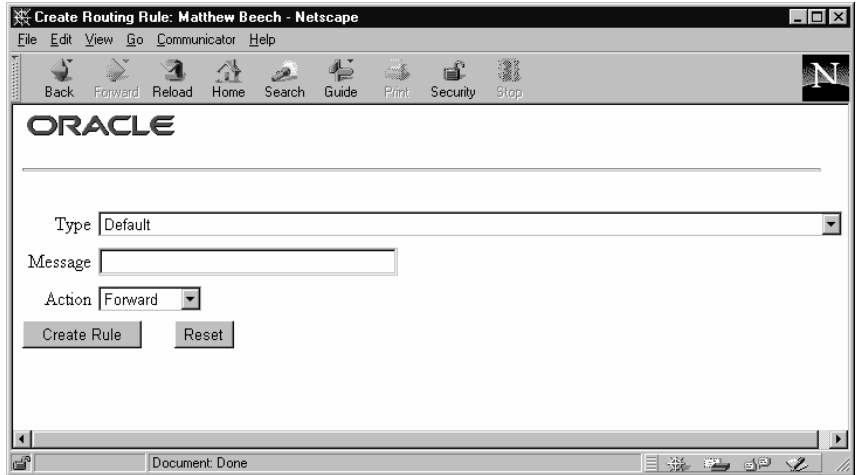


Attention: Both of these URLs access secured pages, so if you have not yet logged on as valid user in the current web session, you will be prompted to do so before the page appears. See: Secure the Workflow Database Connection Descriptor: page 2 – 21.

- The Notification Routing Rules page for the role appears, listing all existing rules for the current role. Choose Create Rule.



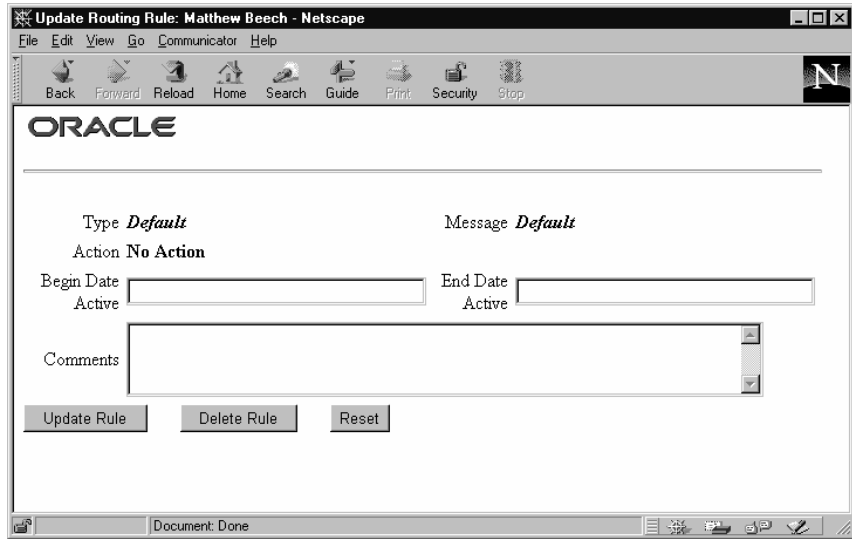
- In the Type poplist field, choose the item type to which this rule applies or choose Default if you want this rule to apply to notifications associated with any item type.



4. In the Message field, enter the internal name or display name of the message to which this rule applies. If Type is specified, but Message is blank, this rule applies to all notifications of the indicated item type. If Type is set as Default and Message is blank, this rule applies to all notification messages.

Note: If the intent of this rule is to automatically respond to a set of messages, then you must specify the name of the item type and message to which this rule applies, as different notification messages require different response values.

5. In the Action poplist field, choose the action that you want this rule to perform:
 - Forward—forward the notification to a designated role.
 - Respond—respond to the message with a set of predefined response values.
 - No Action—leave the notification in the your inbox and do nothing. You can define a rule with this action to exclude a certain subset of notifications from a more encompassing rule. For example, suppose you have a rule that forwards all notification messages to another role, but you want to exclude a subset of notifications from that rule. To accomplish this, you can define a new rule that applies only to that subset of notifications, whose action is No Action.
6. Choose Create Rule to display the Update Routing Rule page. The fields in this page vary depending on your rule's action.

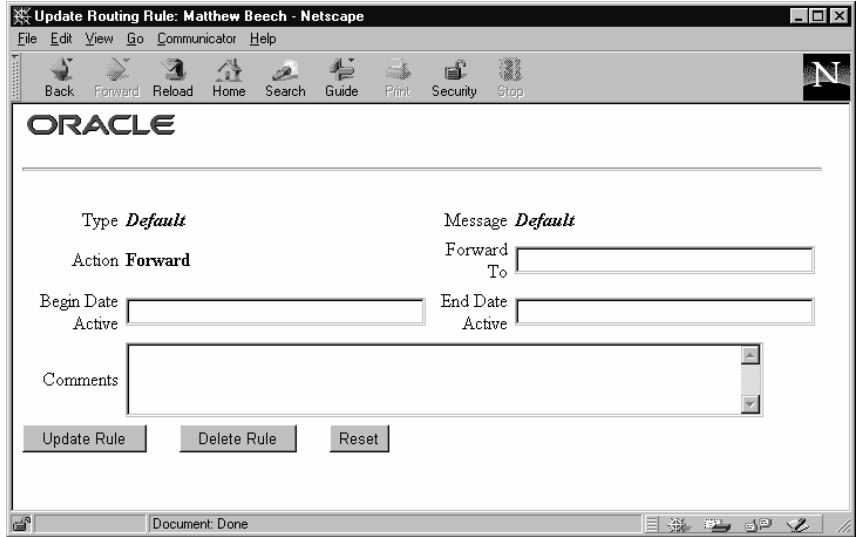


7. Enter values in the Begin Date Active and End Date Active fields to specify the period that this rule should be active. Specify the date, using the default date format of your database. If you leave Begin Date Active blank, the rule is effective immediately. If you leave End Date Active blank, the rule is effective indefinitely.

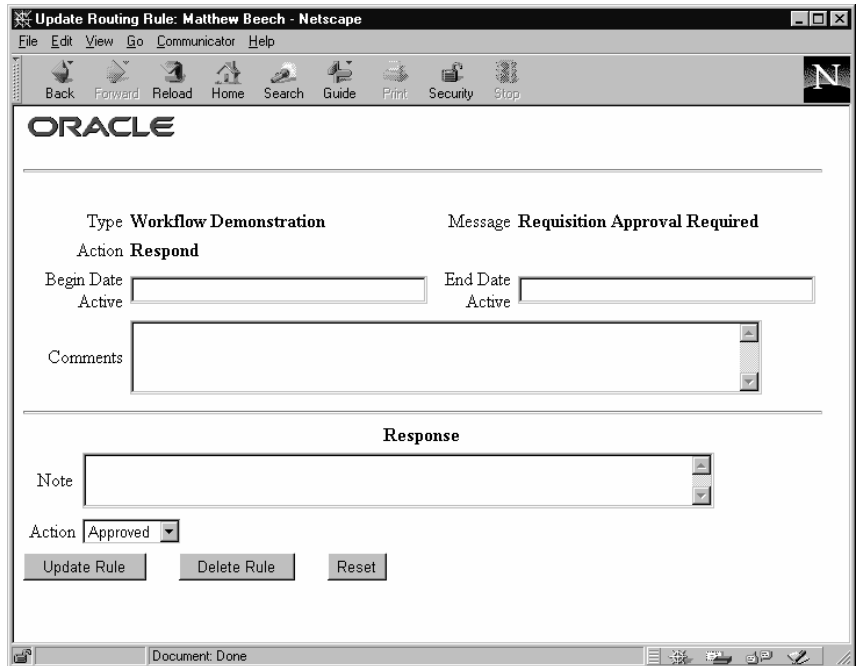


Warning: Since you can define different rules for the same notification(s) to be effective at different times, the Notifications Routing Rules web page does not prevent you from defining multiple rules for the same notification(s). You should be careful to ensure that rules for the same notification(s) do not overlap in their effective dates. If multiple rules are effective for the same notification, the Automatic Notification Handler picks one rule at random to apply.

8. In the Comments field, enter any text that you want to append to the notification when the rule is applied. The comments appear in a special "Prior comments" field.
9. If your rule action is "Forward", enter the role display name that you want to forward the notifications to in the Forward To field that appears.



If your rule action is "Respond", set the response values that you want to automatically reply with in the Response section that appears for that message.



10. Choose Update Rule when you finish defining your rule.
11. Once the rule is processed, choose the [Return to Notification Routing Rules](#) link to display an updated list of your role's routing rules.

► **To Update or Delete an Automatic Notification Routing Rule**

1. Connect to the URL for the Notification Routing Rules web page:

`<webagent>/wf_route.list`

`<webagent>` represents the base URL of the Oracle Web Agent used by Oracle Workflow. It looks something like

`http://<server.com:portID>/<PLSQL_agent_virtual_path>`. See: [Identifying the Oracle Web Agent used by Oracle Workflow: page 2 – 15](#).

2. The Notification Routing Rules page for the role appears. Click on the action for the rule you wish to update or delete.
3. In the Update Routing Rule web page make your changes to the rule and choose Update Rule.

You can also choose Reset before choosing Update Rule to undo your changes and reset the rule back its most recently saved state.

If you wish to delete the rule, choose Delete Rule.

Accessing the Oracle Workflow Home Page

Use the Oracle Workflow home page to link to all of Oracle Workflow's web-based features. This page centralizes your access to the features so you do not have to remember individual URLs.

► To Access the Oracle Workflow Home Page

1. Use a web browser to connect to the URL for the home page:

```
<webagent>/wfa_html.home[?message=<message>]
```

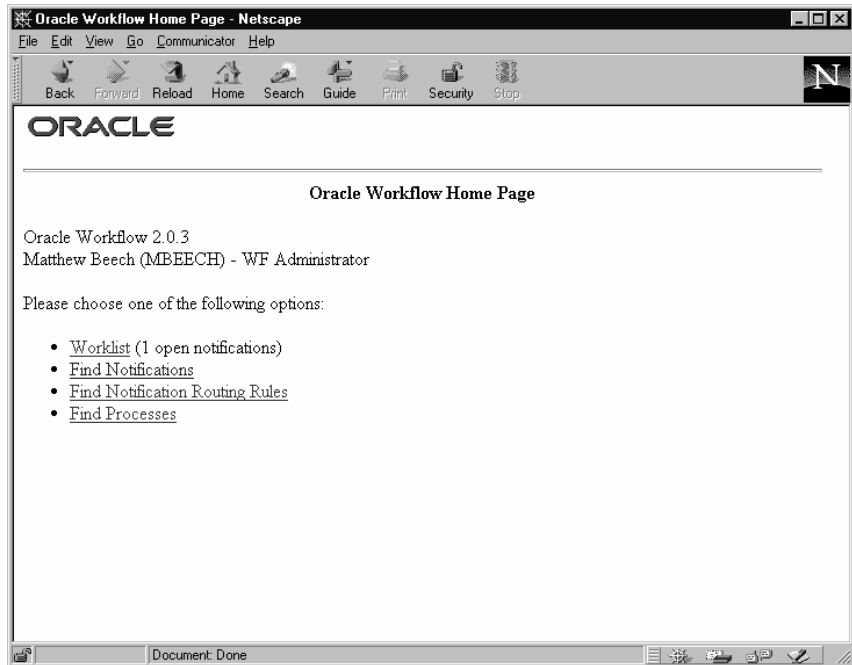
<webagent> represents the base URL of the Oracle Web Agent used by Oracle Workflow. It looks something like `http://<server.com:portID>/<PLSQL_agent_virtual_path>`. See: *Identifying the Oracle Web Agent used by Oracle Workflow: page 2 – 15.*

You can also append an optional message argument (without the square brackets []) where *<message>* represents any message text that you want to appear on the home page above the bulleted list of HTML links. Note that if the message string is more than one word, you need to enter the string as *word1+word2+...+wordn*, where the plus sign (+) represents a space.



Attention: This is 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. See: *Secure the Workflow Database Connection Descriptor: page 2 – 21.*

2. The web page appears as follows:



This page identifies the current version of Oracle Workflow, the role you are currently logged in as, and whether the role has workflow administrator privileges. See: Identifying the Oracle Workflow Administration Role: page 2 – 18.

3. Choose the [Worklist](#) link to access your list of workflow notifications. This link also provides a count of how many open notifications you have. See: Notification Worklist: page 8 – 15.
4. Choose the [Find Notifications](#) link to access the web page that lets you specify criteria to presort and display a subset of your notification worklist. See: Find Notifications: page 8 – 14.
5. Choose the [Notification Routing Rules](#) link to access the web page for defining automatic notification routing rules. If you are logged in as a role with workflow administrator privileges, you get a [Find Notification Routing Rules](#) link that takes you to the web page that locates the routing rules for the role you specify. See: To Define a Rule for Automatic Notification Routing: page 8 – 20.
6. Choose the [Find Processes](#) link to access the web page that lets you query for a list of workflow process instances that match certain search criteria. Once you find a specific process instance, you can view its status details in the Workflow Monitor. See: Using the Find Processes Web Page: page 9 – 9.

CHAPTER

9



Monitoring Workflow Processes

This chapter discusses how to monitor an instance of a workflow process.

Overview of Workflow Monitoring

Once a workflow has been initiated for a work item, it may be necessary to check on its status to ensure that it is progressing forward, or to identify the activity currently being executed for the work item. Oracle Workflow provides an Oracle Applications Workflow Status form (for Oracle Applications users), a Java-based Workflow Monitor tool, and a view called WF_ITEM_ACTIVITY_STATUSES_V to access status information regarding for an instance of a workflow process.

See Also

Oracle Workflow Views: page 7 – 62

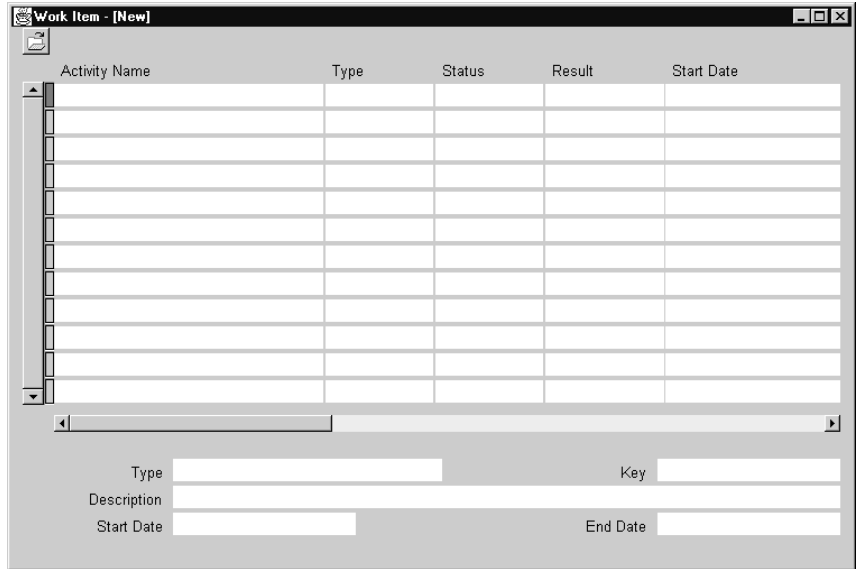
Workflow Status Form

The Workflow Status form is an Oracle Applications form that can be called by any other Oracle Applications form to display status information about an instance of a workflow process.

Your form should use *FND_FUNCTION.EXECUTE* to call the Oracle Workflow Status form function (the developer form name and function name are FNDWFIAS and FND_FNDWFIAS, respectively) and pass it the following parameters:

```
ITEM_TYPE=<item_type> ITEM_KEY=<item_key>
```

Replace *<item_type>* with the internal name of the item type that the workflow process is associated with and replace *<item_key>* with the item key that uniquely identifies the work item in that process. See: *Overview of Menus and Function Security, Oracle Applications Developer's Guide*.



The Workflow Status form is a folder form that displays a list of the activities for an instance of a workflow process and identifies each activity's type, status, result, start date, and end date. See: *Customizing the Presentation of Data in a Folder, Oracle Applications User's Guide*.

To present a process' activities in the order of execution, you should set the Workflow Status folder form to order by `Activity_Start_Date` and `Execution_Time` (already done by default), both in ascending order.

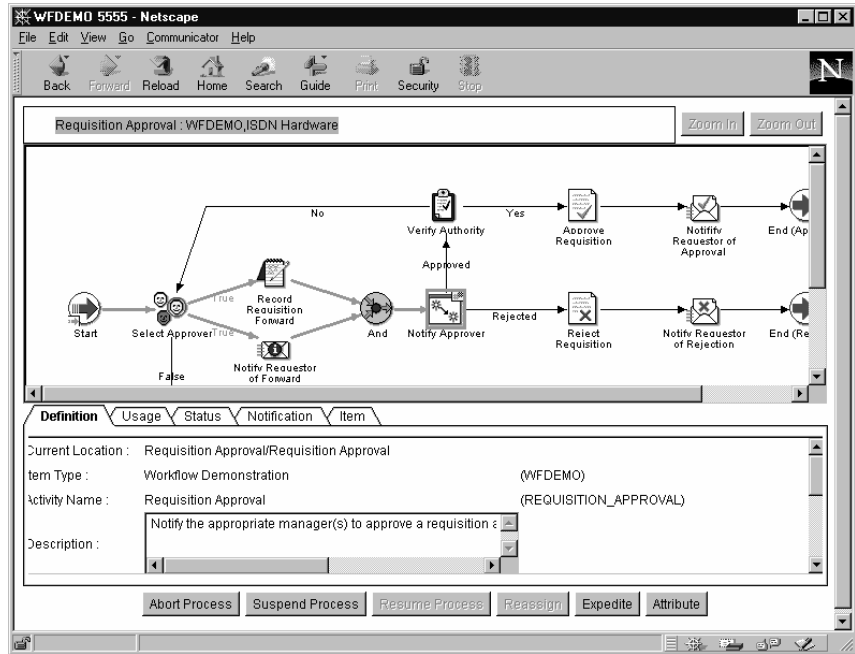
Workflow Monitor

The Workflow Monitor is a tool that allows you to view and administer the status of a specific instance of a workflow process. You can use the point-and-click interface to display detailed status information about activities in the process as well as about the process as a whole. The Workflow Monitor can be run in 'USER' or 'ADMIN' mode, where 'ADMIN' mode provides additional details and functionality pertinent only to a workflow administrator. See: *Workflow Monitor Access: page 9 – 8*.

The Workflow Monitor consists of the following sections:

- Process Title
- Process Diagram Window

- Detail Tab Window
- Administration Buttons



Process Title

The process title appears in the upper left of the Workflow Monitor and displays the name of the workflow process and the name of the item type and user key that uniquely identify a running instance of that process in the process diagram window. If no user has been set, then the item key is displayed instead. If you drill down into a subprocess in the process diagram window, the process title updates to display the subprocess name.

Click on the process title to deselect any selected activity in the process diagram window and to direct the detail tab window to display information about that process or subprocess as a whole.

Process Diagram Window

The process diagram window is a scrolling canvas that displays the diagram of the workflow process or subprocess currently listed in the process title. This diagram is identical to the diagram created in Oracle

Workflow Builder. Note, however, that you cannot use the Workflow Monitor to edit this diagram.

The process diagram window provides graphical cues about the status of the process and its activities:

- An activity icon may be highlighted with a colored box to indicate that it is in an "interesting" state:

Color of Box	State	Possible Status Code
Red	Error	ERROR
Green	Active/In Progress	ACTIVE, NOTIFIED, DEFERRED
Yellow	Suspended	HOLD
<none>	Normal	COMPLETE, WAITING, NULL

Table 9 – 1 (Page 1 of 1)

- Any transition (arrow) that has been traversed appears with a thick green line, while an untraversed transition appears with a thin black line.
- Click on an activity icon in the diagram to select it and update the detail tab window to display information about the activity.
- Click on any empty space in the process diagram to deselect the currently selected activity icon and to refresh the detail tab window to display information about the current process as a whole.
- Double-click on an activity icon that represents a subprocess to drill down to the subprocess' diagram. This action automatically updates the process title to reflect the name of the subprocess and updates the detail tab window to display information about the subprocess as a whole.

Alternatively, you can select the subprocess activity and choose Zoom In to drill down to the subprocess' diagram. Choose Zoom Out to navigate back to the process at the previous level.

Detail Tab Window

The detail tab window, which appears below the process diagram, is a vertically scrollable display area that provides information about a selected process or activity.

The information appears as follows for each tab, where rows preceded by an asterisk (*) or values shown in bold parentheses () appear only when the monitor is run in 'ADMIN' mode:

Definition Tab

Process Display Name: Activity Display Name
Item Type: Item Type display name (internal name)
Activity Name: Activity display name (internal name)
Description: Activity description
Activity Type: Process, Notification, or Function
Message: Message internal name
Function: Name of PL/SQL procedure called by activity
Result Type: Result type display name (internal name)
Timeout: Timeout value in days, hours, minutes
*Cost: Function activity cost in seconds
*Loop Mode: IGNORE or RESET
*Error Process: Error process assigned to activity, if any

Usage Tab

Process Display Name: Activity Display Name
Start/End: No, Start, or End (process result)
Performer: Role name or item attribute internal name
*Comment: Comments for the process activity node

Status Tab

Process Display Name: Activity Display Name
Status: Activity status
Result: Activity result (result code)
Begin Date: Date activity begins
End Date: Date activity ends
*Notification: Notification ID
Assigned User: Role name or item attribute internal name
(shown only if Activity Status is 'ERROR')
*Error Name: Name of error
Error Message: Error message
*Error Stack: Error stack

Notification Tab

Process Display Name: Activity Display Name
*ID: Notification ID
Recipient: Recipient of notification
Status: Notification status
Begin Date: Date notification is delivered
End Date: Date notification is closed

(If the selected activity is a Voting activity, then instead of displaying the above information, this tab displays the results of the vote. This tab displays for each recipient, the notification status and voting result.)

Item Tab

Process Display Name: Item Type, Item Key (or User Key, if set)
Owner: Owner of the item, not implemented yet
Begin Date: Date workflow process instance is created
End Date: Date workflow process instance is completed
<Item Attribute>: <type(format)> <value>
...

Administration Buttons

The administration buttons appear beneath the detail tab window only when the Workflow Monitor is run in 'ADMIN' mode. Each button allows you to perform a different administrative operation by calling the appropriate Workflow Engine API. The buttons and their behavior are as follows:

- **Abort Process**—Available only if you select the process title or a process activity. Calls *WF_ENGINE.AbortProcess* to abort the selected process and cancel any outstanding notifications. Prompts for a result to assign to the process you are about to abort. The process will have a status of Complete, with the result you specify. See: *AbortProcess*: page 7 – 18.
- **Suspend Process**—Available only if you select the process title or a process activity. Calls *WF_ENGINE.SuspendProcess* to suspend the selected process so that no further activities can be transitioned to. See: *SuspendProcess*: page 7 – 16.
- **Resume Process**—Available only if you select a suspended process. Calls *WF_ENGINE.ResumeProcess* to resume the suspended process to normal execution status. Activities that were transitioned to when the process was suspended are now executed. See: *ResumeProcess*: page 7 – 17.
- **Reassign**—Available only if you select a notification activity. Calls *WF_ENGINE.AssignActivity* to reassign a notification activity to a different performer. Prompts for a role name. See: *AssignActivity*: page 7 – 30.
- **Expedite**—Available if you select the process title, or an activity. Calls *WF_ENGINE.HandleError* to alter the state of an errored

activity, or to undo the selected activity and all other activities following it to rollback part of the process. Prompts you to select Skip, to skip the activity and assign it a specified result, or Retry, to reexecute the activity. See: *HandleError*: page 7 – 31.

- **Attribute**—Always available so that you can change the value of an item type attribute. The current values appear for each item type attribute. After changing a value, choose OK to apply the change.

Workflow Monitor Access

You can control a user's access to the Workflow Monitor in one of two ways. You can either depend on the workflow-enabled application to control access to the Workflow Monitor or provide direct access to the Find Processes web page.

Application-controlled Access to the Workflow Monitor

Identify within the logic of your application code, the workflow process instance(s) that a user is allowed to view, and whether to run the monitor in 'ADMIN' or 'USER' mode for that user. You must also provide a means in your application's user interface to call a web browser application that supports Java 1.1.4 and AWT and pass it the Workflow Monitor URL that gets returned from the function *WF_MONITOR.GeDiagramURL()*. The returned URL will have a hidden password attached that provides the user access to the Workflow Monitor in either 'ADMIN' or 'USER' mode. See: *GetDiagramUrl*: page 7 – 58.

Note: In Oracle Applications, you can call the function *FND_UTILITIES.OPEN_URL* to open a web browser and have it connect to a specified URL. See: *FND_UTILITIES:Utility Routine*, *Oracle Applications Developer's Guide*.

Provide Access to the Find Processes Web Page

Another way to access the Workflow Monitor is to pass the Find Processes URL to a web browser that supports Java 1.1.4 and AWT. The Find Processes page requires user authentication before access. Depending on whether Oracle Workflow is configured to use Oracle Self-Service Web Applications or Oracle WebServer security, the user must log in using the appropriate username and password if he or she has not done so for the current browser session. If the user has

workflow administrator privileges, the Find Processes web page that appears lets the user search for any workflow process instance. If the user does not have workflow administrator privileges, the Find Processes web page lets the user search for only processes that the user owns, as set by a call to the Workflow Engine *SetProcessOwner* API. The user can then display the notifications or the process diagram for any of those process instances in the Workflow Monitor. .

The Find Processes URL looks similar to the following:

```
<webagent>/wf_monitor.find_instance
```

<webagent> is the web agent string that you can retrieve from the WF_WEB_AGENT token in the WF_RESOURCES table by calling WF_CORE.TRANSLATE(). See: Identifying the Oracle Web Agent used by Oracle Workflow: page 2 – 15 and TRANSLATE: page 7 – 41.

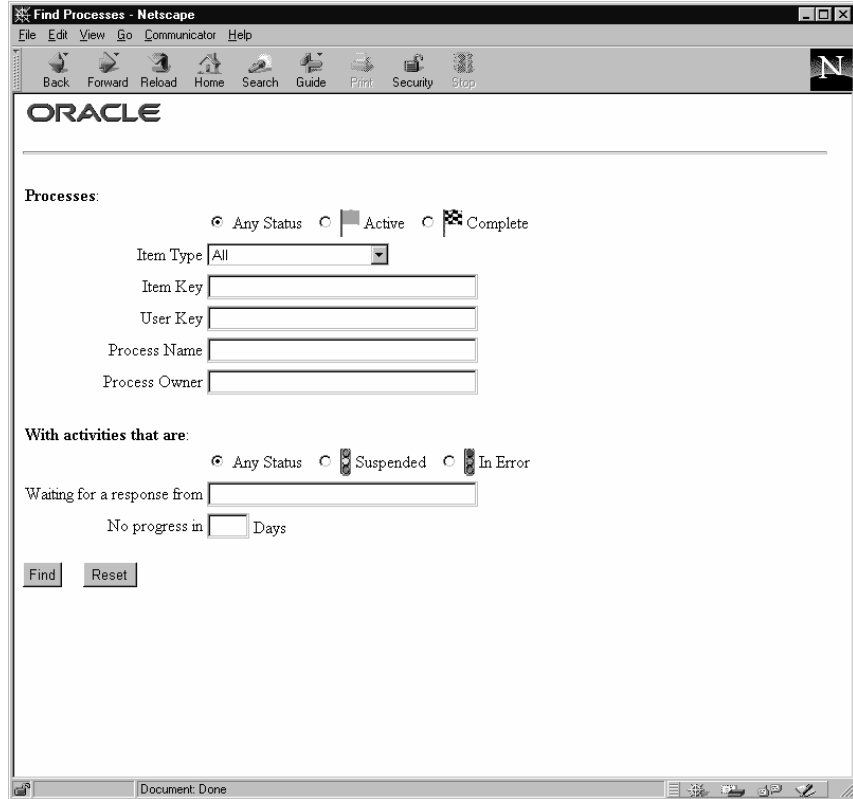
Note: In Oracle Applications, you can call the function *FND_UTILITIES.OPEN_URL* to open a web browser and have it connect to a specified URL. See: FND_UTILITIES:Utility Routine, *Oracle Applications Developer's Guide*.

Note: You can also access the Find Processes web page from the Oracle Workflow home page. See: Accessing the Oracle Workflow Home Page: page 8 – 27

Using the Find Processes Web Page

The Oracle Workflow Find Processes web page allows you to query for a list of workflow process instances that match certain search criteria.

► **To Specify Search Criteria in the Find Processes Page**



1. You can enter search criteria using any combination of the following fields to find a subset of workflow process instances:
 - Process Status—Specify Any Status, Active, or Complete.
 - Item Type—Select the item type of the workflow process instances you want to find, or select All to find workflow process instances for all item types.
 - Item Key or User Key—Specify an item key or a user key. An item key presents the internal identifier of an item whereas a user key is an end-user identifier assigned to an item. A user key may not necessarily be unique to an item. See: [SetItemUserKey](#); page 7 – 11.
 - Process Name—Specify the display name of a process activity.

2. If you log on as a user with Workflow Administrator privileges, you can search for and display any process instance, even if you do not own the process. In the Process Owner field, enter the internal name of any role defined in WF_ROLES to list only processes owned by that role. Alternatively, leave the field blank to list all process instances that match your search criteria regardless of the process owner.

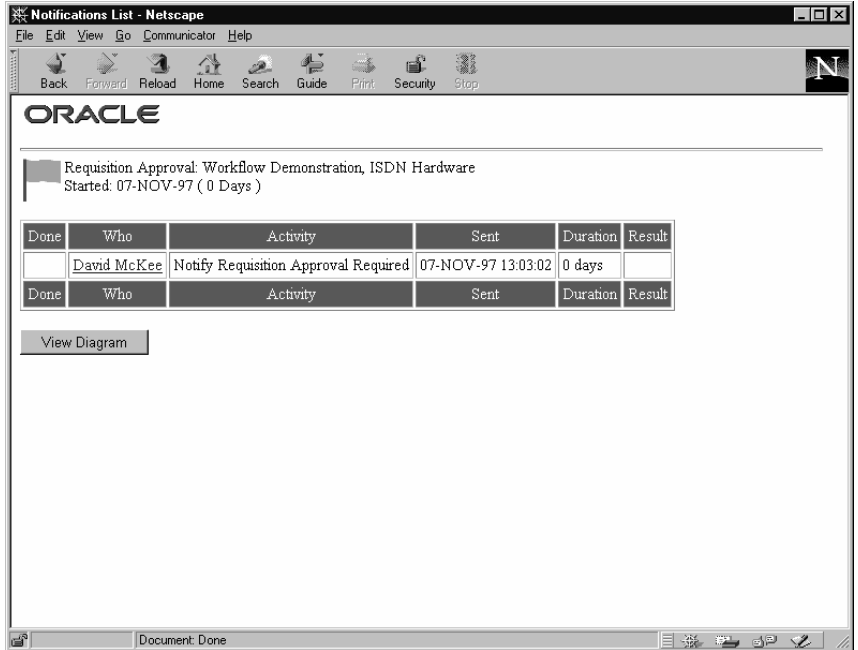
If you do not have Workflow Administrator privileges, then the Process Owner field reflects the internal name of the role you are logged in as for the current web session and you are allowed to search and display only processes that you initiated or are the primary participant of.

Note: You can set the owner of a process by making a call to the WF_ENGINE.SetItemOwner API. The owner of a process is the person who initiated the process or is the primary participant of the process.

3. Optionally, you can also find workflow process instances with activities that are Suspended, In Error, or that have Any Status.
4. You can find workflow process instances that have activities waiting for a response from a particular user or role.
5. You can also identify workflow process instances that have not progressed for a specified number of days.
6. When you finish entering your search criteria, choose Find to find all matching process instances.

Item Type	Item Key	User Key	Process Name	Complete	In Error	Suspended	Begin Date
Workflow Demonstration	19898	WF-101 Course Enrollment	Requisition Approval				03-NOV-97 23:30:23
Workflow Demonstration	19899	HW-200	Requisition Approval	☑			03-NOV-97 23:32:29
Workflow Demonstration	19900	SW-200	Requisition Approval			⏸	03-NOV-97 23:32:54
Workflow Demonstration	5555	ISDN Hardware	Requisition Approval				07-NOV-97 13:03:02
Item Type	Item Key	User Key	Process Name	Complete	In Error	Suspended	Begin Date

7. The Process List provides a summary of all workflow process instances that match your search criteria.
8. Choose the Process Name of a particular row in the summary to display the list Notifications List for the process. The Notifications List shows all the current notifications that have been sent for a process. The list also describes what each notification activity is, who it is assigned to, when it was sent, whether it has been completed, how many days have passed before completion, and the result of the activity.



9. You can choose the user in the Who column to send email to the user that the notification has been assigned to or choose the View Diagram button to display the process in the Workflow Monitor. If you connected to the session as a user with Workflow Administrator privileges, the Workflow Monitor displays the process in 'ADMIN' mode, otherwise the process is displayed in 'USER' mode. See: Workflow Monitor: page 9 – 3.

See Also

Setting Up an Oracle Workflow directory service: page 2 – 7

Identifying the Oracle Workflow Administration Role: page 2 – 18

CHAPTER

10

Sample Workflow: Requisition Approval Process

This chapter examines a sample workflow process definition called Requisition Approval. Some of the components in the process, such as item type attributes, messages, functions, and notifications, are discussed in detail to illustrate how they are integrated into the process.

Requisition Approval Process

The Requisition Approval process is an example of a workflow process that gets initiated when you create a new requisition to purchase an item. The Requisition Approval 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 approved.

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. The files necessary to setup and run this demonstration are not provided with the version of Oracle Workflow embedded in Oracle Applications.



Attention: For detailed information about runnable workflow processes that are integrated with Oracle Applications or Oracle Self-Service Web Applications, refer to the appropriate Oracle Applications User's Guide or online documentation. See: Predefined Workflow Embedded in Oracle Applications and Oracle Self-Service Web Applications: page B - 2.



Attention: Oracle Self-Service Web Applications provides a predefined Requisition Approval 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.

To run this sample workflow you must install 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 Approval 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 Approval process in detail to give you an understanding of what each activity in this workflow accomplishes.

Installing the Demonstration Data Model

The demonstration data model for the sample Requisition Approval process is available for installation only for the standalone version of Oracle Workflow. The demonstration data model is installed using Oracle Installer. Oracle Installer asks if you wish to install the demonstration data model when you install the Oracle Workflow server. The files used in the installation are copied to the *demo* or *demo/<language>* subdirectories of your Oracle Workflow server directory structure.



Attention: For the Requisition Approval process demonstration to work properly, you should perform the steps required to set up Oracle Workflow after you install the demonstration data model. See: *Implementing Oracle Workflow*: page 2 – 4.

The installation does the following:

- 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 userids listed in ascending order with the employee having the most authority listed last: JANDERSON, DMCKEE, MBEECH, JRUSH, and JSMITH.

- WF_REQDEMO_EMP_AUTHORITY—maintains the spending limit for each employee. The limit for each employee follows the employee’s userid: JANDERSON:500, DMCKEE:1000, MBEECH:2000, JRUSH:3000, JSMITH:4000.
- The script *wfdemoc.sql* also inserts the following seed data into the WF_LOCAL_USERS, WF_LOCAL_ROLES, WF_LOCAL_USER_ROLES tables:

Users	Roles			
	ADMIN	MANAGERS	WORKERS	OTHERS
SYSADMIN	x			
WFADMIN	x			
JANDERSON			x	
DMCKEE			x	
MBEECH			x	x
JRUSH		x		x
JSMITH		x		

Table 10 – 1 (Page 1 of 1)

Each of these users have their E-mail address assigned to 'WFTEST'. This allows you to use one mail account to check/test their notifications. You can also change each user’s E-mail address to anything you like by editing the *wfdemoc.sql* script and rerunning it.

Also every user except JANDERSON has a Notification Preference of 'MAILHTML', which allows them to get individual notifications via E-mail if a Notification Mailer is set up. JANDERSON has a Notification Preference of 'SUMMARY', which allows him to receive a periodic E-mail summarizing all the notifications he has received if an appropriate Notification Mailer is set up. In all cases, users can also always query for their notifications using a Web browser by connecting to the Oracle Workflow Notification Web page.



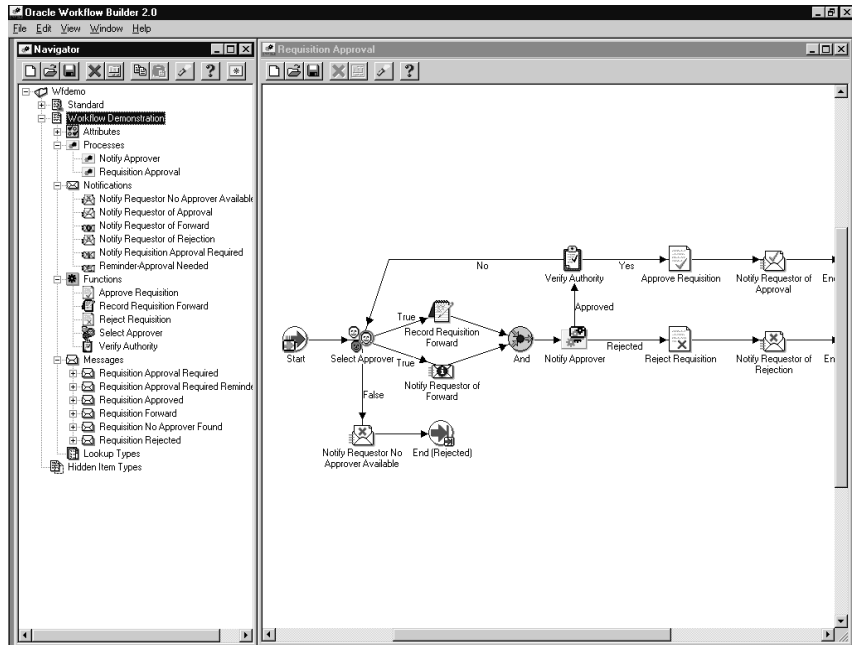
Attention: Your Oracle Workflow directory service must include the WF_LOCAL_USERS, WF_LOCAL_ROLES and WF_LOCAL_USER_ROLES tables for the *wfdemoc.sql* script to

run properly. See: Setting Up an Oracle Workflow Directory Service: page 2 – 7.

- Calls the scripts *wfdemos.sql* and *wfdebob.sql* to create the PL/SQL spec and body for a package called WF_REQDEMO. This package contains:
 - The PL/SQL stored procedures called by the function activities used in the Requisition Approval 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 Approval 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 Approval process demonstration.
- Loads the Requisition Approval Process workflow definition from *wfdemo.wft* into the database. You can view this process in Oracle Workflow Builder.

Displaying a Process Window

Once you install the demonstration environment, you can view the sample processes in a Process window using Oracle Workflow Builder.



► **To Display the Sample Process in Oracle Workflow Builder**

1. Choose Open from the File menu, and connect to the database where you installed the demonstration data model.

Alternatively, you can connect to the workflow definitions file *wfdemo.wft*, located in the Oracle Workflow *wf20/data/<language>* subdirectory on your PC.

2. Expand the data source, then the Workflow Demonstration item type branch within that data source.
3. Expand the Processes branch within Workflow Demonstration then double-click on a process activity to display the diagram of the process in a Process window.

The Workflow Demonstration Item Type

The Requisition Approval process is associated with an item type called Workflow Demonstration. This item type identifies all demonstration workflow processes available. Currently there are two workflow processes associated with Workflow Demonstration: Requisition Approval and Notify Approver.

If you examine the property page of Workflow Demonstration, you 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 as the process to run.

The Workflow Demonstration 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.

Workflow Demonstration Item Type Attributes

Display Name	Description	Type	Length/Format/ Lookup Type
Forward From Display Name	Name of the person that the requisition is forwarded from	Text	
Forward From Username	Username of the person that the requisition is forwarded from	Text	30
Forward To Display Name	Name of the person that the requisition is forwarded to	Text	
Forward To Username	Username of the person that the requisition is forwarded to	Text	30
Requestor Display Name	Name of the requisition preparer	Text	
Requestor Username	Username of the requisition preparer	Text	30
Requisition Amount	Requisition amount	Number	9,999,999,999.99
Requisition Number	Unique identifier of a requisition	Text	
Requisition Description	Unique user identifier of a requisition	Text	30
Requisition Process Owner	Username of the requisition owner	Text	30
Note	Note	Text	
Monitor URL	Monitor URL	URL	

Table 10 – 2 (Page 1 of 1)

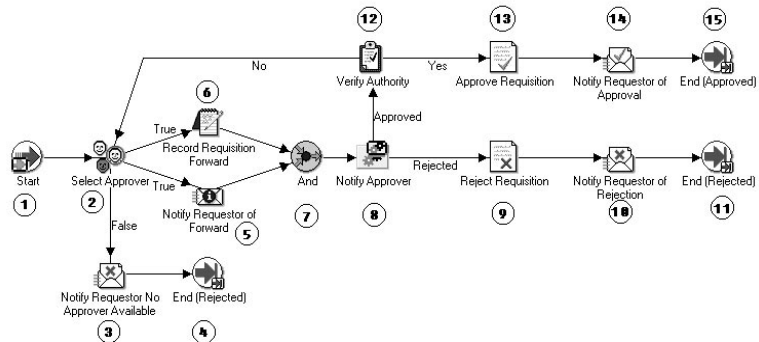
Summary of the Requisition Approval Process

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 Approved or Rejected (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 by making calls to the Workflow Engine *CreateProcess* and *StartProcess* APIs.

The Details property page of the process activity indicates that the Requisition Approval process has an error process called WFDEMO_ERROR_PROCESS associated with it, which gets initiated only when an error is encountered in the process. 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 WFDEMO_ERROR_PROCESS, which is the Demonstration Error Process. The Demonstration Error Process is associated with the System:Error item type. Currently the process simply executes the standard Default Error Notification activity to provide information associated with the error. You can customize the process further to suit your needs. See: Default Error Process: page 5 – 16.

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.



The Requisition Approval workflow begins when you run a script called *wfrundemo.sql* or submit a requisition using the Requisition Approval 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 Approval Workflow: page 10 – 18.

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 Rejected. 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 Rejected.

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 Approved.

Requisition Approval 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. All function activities execute PL/SQL stored procedures which you must create and store in the Oracle RDBMS. The naming convention for the PL/SQL stored procedures used in the Requisition Approval process is:

`WF_REQDEMO . <PROCEDURE>`

`WF_REQDEMO` is the name of the package that groups all the procedures used by the Requisition Approval 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 – 26 and Example Notification Activities: page 10 – 32.

Start (Node 1)

This is a Standard function activity that simply marks the start of the process.

Function	<i>WF_STANDARD.NOOP</i>
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 imaginary 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	<i>WF_REQDEMO.SelectApprover</i>
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.

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.

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, however, if you have a Purchasing/Requisition application that you wish to integrate this workflow into, 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	<i>WF_REQDEMO.Forward_Req</i>
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	<i>WF_STANDARD.ANDJOIN</i>
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, however, if you have a Purchasing/Requisition application that you wish to integrate this workflow into, 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.

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, however, if you have a Purchasing/Requisition application that you wish to integrate this workflow into, 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	<i>WF_REQDEMO.Approve_Req</i>
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.

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)

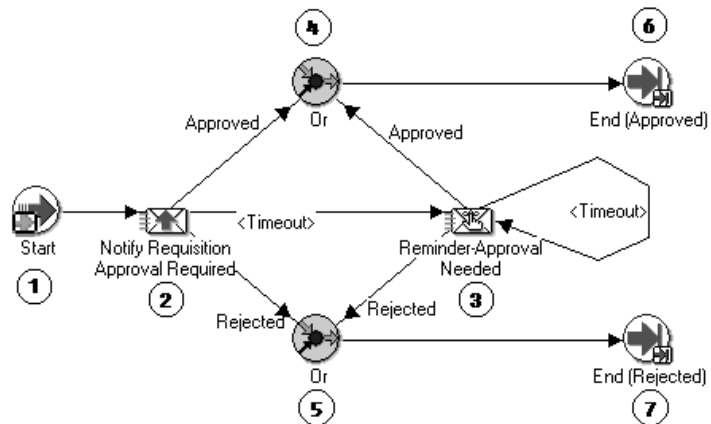
This function activity marks the end of the process. Although the activity itself does not have a result type, each node of this activity in the process must have a process result assigned to it. The process result is assigned in the property page of the activity node. Since the Requisition Approval 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	<i>WF_STANDARD.NOOP</i>
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 Approved or Rejected (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 5 unique activities, several of which are reused to comprise the 7 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.



The subprocess begins at node 1 with the Start activity. At node 2, the process notifies the approver to approve a requisition within a specified period of time. If the approver approves the requisition, the subprocess ends at node 6 and returns the result Approved to the top level Requisition Approval process. Similarly, if the approver rejects the requisition, the subprocess ends at node 7 and returns the result Rejected to Requisition Approval process.

If the approver does not respond by the due date, the subprocess takes the <Timeout> transition to node 3 to send a reminder to the approver

to approve the requisition. Node 3 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 3 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 6 or 7, 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 None

Activities

Notify Requisition Approval Required (Node 2)

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 message includes two "Respond" attributes that prompt the approver for responses. One is called Action and it provides the approver with the values 'APPROVE' or 'REJECT' from the lookup type called Approval. Action has an internal name of Result, which indicates that the value that the approver selects becomes the result that determines which activity branch the Workflow Engine transitions to next.

The other "Respond" attribute is called Note and this attribute prompts the approver for any additional comments that he/she wants to include in the notification response regarding the requisition review.

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 3)

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 message includes two "Respond" attributes that prompt the approver for responses. One is called Action and it provides the approver with the values 'APPROVE' or 'REJECT' from the lookup type called Approval. Action has an internal name of Result, which indicates that the value that the approver selects becomes the result that determines which activity branch the Workflow Engine transitions to next.

The other "Respond" attribute is called Note and this attribute prompts the approver for any additional comments that he/she wants to include in the notification response regarding the requisition review.

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 4 and 5)

This Standard function activity merges two or more parallel branches in a flow as soon as an activity in any one of those branches complete.

Function	<i>WF_STANDARD.ORJOIN</i>
Result Type	None
Prerequisite Activities	None

End (Nodes 6 and 7)

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	<i>WF_STANDARD.NOOB</i>
Result Type	None
Prerequisite Activities	Start

Initiating the Requisition Approval Workflow

You can use either of the following methods to initiate the Requisition Approval workflow:

- Run the script *wfrundemo.sql*.
- Submit a requisition using the Requisition Approval Demonstration web page.

You can also create your own custom end-user application interface to let users create requisitions that automatically initiate the Requisition Approval 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 Approval process. See: Sample StartProcess Function: page 10 – 22.

► **To Run *wfrundemo.sql***

1. Enter the following command to run the script *wfrundemo.sql* in SQL*PLUS:

```
sqlplus <username>/<password>@<alias> @wfrundemo.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.

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: page 8 – 6* and *Viewing Notifications from a Web Browser: page 8 – 12*.

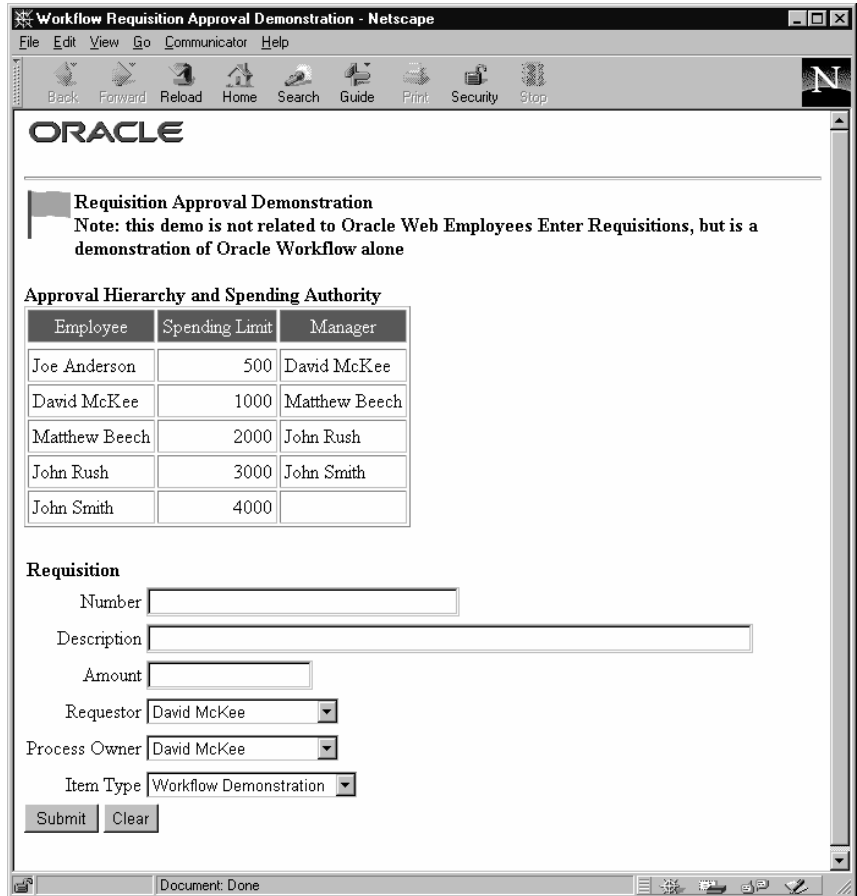
You can also access the Workflow Monitor to view the status of the workflow process. See: *Using the Find Processes Web Page: page 9 – 9*.

► **To Use the Requisition Approval Demonstration Web Page**

1. Enter the following URL in a web browser to access the Requisition Approval Demonstration web page:

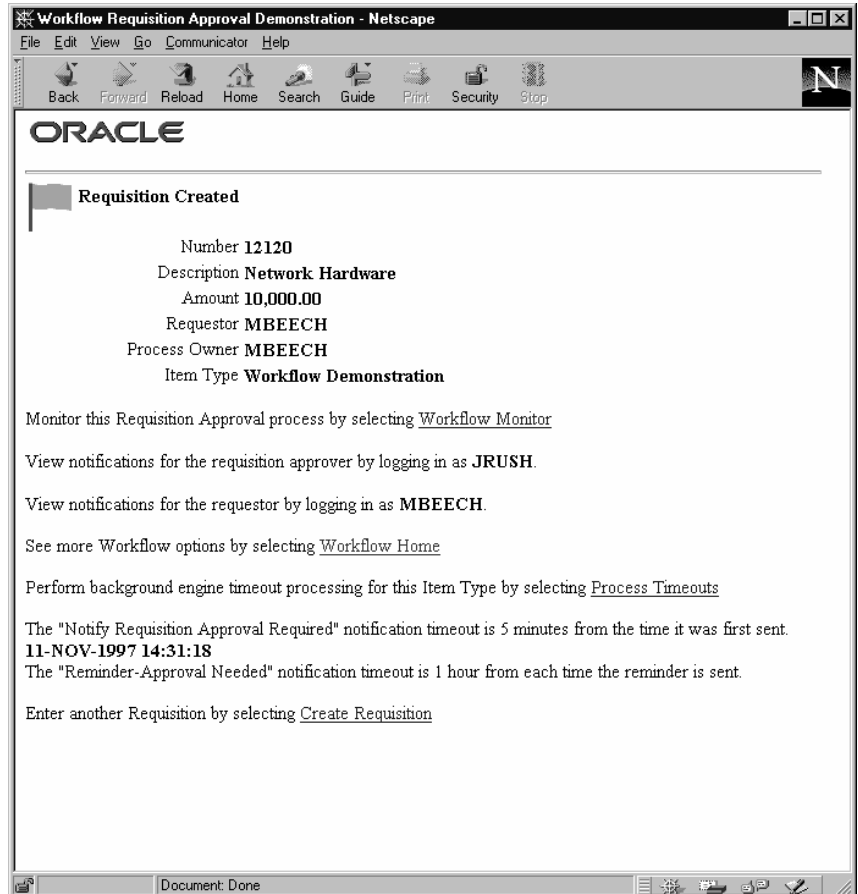
```
<webagent>/wf_reqdemo.create_req
```

`<webagent>` represents the base URL of the Oracle Web Agent used by Oracle Workflow. It looks something like `http://<server.com:portID>/<PLSQL_agent_virtual_path>`. See: *Identifying the Oracle Web Agent used by Oracle Workflow: page 2 – 15*.



2. The Approval Hierarchy and Spending Authority table summarizes the contents of the demonstration data model. Following the table are the input fields you can use to submit a requisition.
3. Enter a unique requisition number.
4. Specify a unique requisition description of 80 characters or less.
5. Enter a requisition amount. The amount should be a number without formatting.
6. 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.

7. Choose submit to initiate the Requisition Approval process and to navigate to the Requisition Approval Demonstration 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 HTML links to other Oracle Workflow web components:
 - **Workflow Monitor**—this link navigates to the Workflow Monitor Notifications List where you can choose View Diagram to display the process diagram for the requisition you submitted in ADMIN mode. See: Workflow Monitor: page 9 – 3.
 - **Workflow Home**—this link navigates to the Oracle Workflow Home page for your current role, where you can access links to

view and respond to the notifications for the process. See: Accessing the Oracle Workflow Home Page: page 8 – 27.



Attention: Both the Workflow Monitor and Workflow Home links navigate to secured pages, so if you have not yet logged on as a valid role in the current web session, you will be prompted to do so before either page appears. See: Secure the Workflow Database Connection Descriptor: page 2 – 21.



Attention: To log in as a different user in a web session, you must first exit your current web browser session, then restart the web browser and login again as the other user.

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 Approval Demonstration web page.

Sample *StartProcess* Function

Both *wfrundemo.sql* and the Requisition Approval Demonstration web page call a PL/SQL stored procedure named *WF_REQDEMO.StartProcess* to initiate the Requisition Approval 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 and arrows 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(30) := 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    => 'REQUESTOR_DISPLAY_NAME',
                                avalue   => wf_directory.GetRoleDisplay-
                                Name(RequestorUsername));
10→ wf_engine.SetItemAttrText    (itemtype => itemtype,
                                itemkey  => itemkey,
                                aname    => 'FORWARD_TO_USERNAME',
                                avalue   => RequestorUsername);
11→ wf_engine.SetItemAttrText    (itemtype => itemtype,
                                itemkey  => itemkey,
                                aname    => 'FORWARD_TO_DISPLAY_NAME',
                                avalue   => wf_directory.GetRoleDisplay-
                                Name(RequestorUsername));
12→ wf_engine.SetItemAttrText    (itemtype => itemtype,
                                itemkey  => itemkey,
                                aname    => 'REQUISITION_PROCESS_OWNER',

```

```

13→          wf_engine.SetItemAttrText          avalue => ProcessOwner);
                                                (itemtype => itemtype,
                                                itemkey  => itemkey,
                                                aname   => 'MONITOR_URL',
                                                avalue  => wf_monitor.GetUrl
(wf_core.translate('WF_WEB_AGENT'),
itemtype, itemkey, 'NO'));
14→          wf_engine.SetItemOwner            (itemtype => itemtype,
                                                itemkey  => itemkey,
                                                owner   => ProcessOwner);
15→          wf_engine.StartProcess             (itemtype => itemtype,
                                                itemkey  => itemkey );
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 *wfrundemo.sql* script or to the field values entered in the Requisition Approval 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: page 7 – 8.

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 Workflow Demonstration item type. The argument *ItemKey* is the value of the *RequisitionNumber* parameter that is passed to the *StartProcess* procedure.

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 Approval process, whose internal name is 'WFDEMO', and is identified by the item type and item key that is supplied. See: *CreateProcess*: page 7 – 9.

Note: If you do not pass a value for *<process_int_name>* to the *wfrundemo.sql* script, the selector function for the Workflow Demonstration 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 Approval 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*: page 7 – 11.

5, 6, 7, 8, 9, 10, 11, 12, and 13→ These sections call either the *SetItemAttributeText* or *SetItemAttributeNumber* Workflow Engine API's to set values for the item type attributes defined for this process. The attributes are REQUISITION_NUMBER, REQUISITION_DESCRIPTION, REQUISITION_AMOUNT, REQUESTOR_NAME, REQUESTOR_DISPLAY_NAME, FORWARD_TO_USERNAME, FORWARD_TO_DISPLAY_NAME, REQUISITION_PROCESS_OWNER, and MONITOR_URL, respectively. See: *SetItemAttribute*: page 7 – 22.

14→ This section calls the *SetItemOwner* Workflow Engine API to mark the new runtime instance of the Requisition Approval process with a process owner user name. Users can query for process instances by process owner. See: *SetItemOwner*: page 7 – 13.

15→ This section calls the Oracle Workflow Engine *StartProcess* API to invoke the Requisition Approval process for the item type and item key specified. See: *StartProcess*: page 7 – 14.

Example Function Activities

In general, function activities 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 Approval process in the *demo* subdirectory of the Oracle Workflow directory structure on your server.

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(30);
   l_forward_to_username          varchar2(30);
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→ wf_engine.SetItemAttrText (itemtype => itemtype;
                             itemkey => itemkey,
                             aname => 'FORWARD_FROM_DISPLAY_NAME';
                             avalue => wf_directory.GetRoleDisplay-
                             Name(l_forward_from_username));
7→ l_forward_to_username := wf_reqdemo.GetManager( l_forward_from_username);
8→ wf_engine.SetItemAttrText (itemtype => itemtype;
                             itemkey => itemkey,
                             aname => 'FORWARD_TO_USERNAME';
                             avalue => l_forward_to_username);
9→ wf_engine.SetItemAttrText (itemtype => itemtype;
                             itemkey => itemkey,
                             aname => 'FORWARD_TO_DISPLAY_NAME';
                             avalue => wf_directory.GetRoleDisplay-
                             Name(l_forward_to_username));
10→ if (l_forward_to_username is null) then
        resultout := 'COMPLETE:F';
    else
        resultout := 'COMPLETE:T';
    end if;
11→ end if;
12→ if (funcmode = 'CANCEL') then

```

```

        resultout := 'COMPLETE';
        return;
    end if;
13→ if (funcmode = 'TIMEOUT') then
        resultout := 'COMPLETE';
        return;
    end if;
14→ exception
        when others then
            wf_core.context('WF_REQDEMO', 'SelectorApprover', itemtype,
                itemkey, actid, funcmode);
            raise;
15→ 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*: page 7 – 23.

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 Directory Service *GetRoleDisplayName* API to get the display name of the username stored in `l_forward_from_username` and assigns that name to the `FORWARD_FROM_DISPLAY_NAME` item type attribute by calling the Workflow Engine *SetItemAttrText* API.

7→ 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`.

8→ 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.

9→ This section calls the Directory Service *GetRoleDisplayName* API to get the display name of the username stored in `l_forward_to_username` and assigns that name to the `FORWARD_TO_DISPLAY_NAME` item type attribute by calling the Workflow Engine *SetItemAttrText* API.

10→ 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`.

11→ This ends the check on `funcmode = 'RUN'`.

12→ If the value of `funcmode` is `CANCEL`, then assign `resultout` to be `COMPLETE`.

13→ If the value of `funcmode` is `TIMEOUT`, then assign `resultout` to be `COMPLETE`.

14→ This section calls `WF_CORE.CONTEXT` if an exception occurs.

15→ The `SelectApprover` procedure ends.

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

'p_' to identify parameters that are passed to another procedure and
'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(30);
   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*: page 7 – 23.

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.

8→ The `VerifyAuthority` procedure ends.

Example Notification Activity

The Requisition Approval 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.

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, 'APPROVED' or 'REJECTED' are defined in a lookup type called Approval. These values are defined by a message attribute called Action, whose internal name is RESULT. 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 your
 approval

Body The following requisition requires
 your approval:

 Requisition Number:
 &REQUISITION_NUMBER

Requisition Description:
&REQUISITION_DESCRIPTION
Requisition Amount:
&REQUISITION_AMOUNT
Forwarded From:
&FORWARD_FROM_DISPLAY_NAME
Requestor: &REQUESTOR_DISPLAY_NAME
Note: &NOTE

Item type attributes, preceded by an ampersand '&' in the subject line and 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 item type attributes used in the subject line and body of the message have to be defined as message attributes with a source of 'Send'.

Other message attributes are also defined for this message:

Action This message attribute has a source of 'Respond' and prompts the approver for a response. By giving this attribute an internal name of 'RESULT', the response returned by the approver becomes the result that determines which activity the process transitions to next.

This message attribute is of the type 'Lookup' indicating that the approver must respond from a list of possible values stored in the lookup type specified. In this case the lookup type is called Approval and contains the lookup codes 'APPROVED' and 'REJECTED'.

Note This message attribute has a source of 'Respond' and is of the type 'Text'. This attribute prompts the approver to enter additional 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 Approval process by the activity called Select Approver.

CHAPTER

11

Workflow Administration Scripts

This chapter describes the SQL scripts that workflow administrators can run against an Oracle Workflow server installation.

Miscellaneous SQL Scripts

You can use any of the following administrative scripts to help set up and maintain various features in Oracle Workflow. For the standalone version of Oracle Workflow, the scripts are located on your server in the Oracle Workflow *admin/sql* subdirectory. For the version of Oracle Workflow embedded in Oracle Applications, the scripts are located in the *sql* subdirectory under *\$FND_TOP*.

- Update translation tables—WFNLADD.sql: page 11 – 2.
- Start a background engine—wfbkg.sql: page 11 – 3.
- Show activities deferred for the next background engine execution—wfbkgchk.sql: page 11 – 3.
- Show a notification’s status—wfntfsh.sql: page 11 – 4.
- Reset the protection level for objects—wfprot.sql: page 11 – 4.
- Handle errored activities—wfretry.sql: page 11 – 4.
- Remove data from Oracle Workflow tables
 - wfrmall.sql: page 11 – 5.
 - wfrmitms.sql: page 11 – 5.
 - wfrmitt.sql: page 11 – 5.
 - wfrmtime.sql: page 11 – 6.
 - wfrmita.sql: page 11 – 6.
- Run a workflow process—wfrun.sql: page 11 – 6.
- Display a status report for an item
 - wfstatus.sql: page 11 – 7.
 - wfstat.sql: page 11 – 7.
- Display the version of the Oracle Workflow server—wfver.sql: page 11 – 7.
- Check the directory service data model—wfdirchk.sql: page 11 – 7.

WFNLADD.sql

If you define a new language in your Oracle installation, use WFNLADD.sql to add the missing rows for that language to the Oracle

Workflow translation tables. See: Creating the WF_LANGUAGES View: page 2 – 13.

Use the script as follows:

```
sqlplus <user/pwd> @WFNLADD
```

Wfbkg.sql

If you are using the standalone version of Oracle Workflow, you can use wfbkg.sql to start a background engine. This script calls the WF_ENGINE Background API to run a background engine for the indicated number of minutes.

Use the script as follows:

```
sqlplus <user/pwd> @wfbkg <minutes> <seconds>
```

Replace *<minutes>* with the number of minutes you want the background engine to run, and replace *<seconds>* with the number of seconds you want the background engine to wait between queries.

See Also

Background: page 7 – 19

Setting up Background Workflow Engines: page 2 – 43

Wfbkgchk.sql

Use wfbkgchk.sql to get a list of all activities waiting to be processed by the background engine the next time it runs.

Use the script as follows:

```
sqlplus <user/pwd> @wfbkgchk
```

See Also

Background: page 7 – 19

Setting up Background Workflow Engines: page 2 – 43

Wfntfsh.sql

Use wfntfsh.sql to display status information about a particular notification, given its notification ID.

Use the script as follows:

```
sqlplus <user/pwd> @wfntfsh <notification_id>
```

Wfprot.sql

Use wfprot.sql to reset the protection level of all objects associated with a specified item type.



Attention: If you reset the protection level for all objects in an item type, then none of those objects in the item type will be customizable by users operating at an access level higher than the new protection level.

Use the script as follows:

```
sqlplus <user/pwd> @wfprot <item_type> <protection_level>
```

Replace *<Item_type>* with the item type that you want to reset the protection level for, and replace *<protection_level>* with the new protection level.

Wfretry.sql

Use wfretry.sql to display a list of activities that have encountered an error for a given process instance and then specify whether to skip, retry, or reset any one of those errored activities.

Use the script as follows:

```
sqlplus <user/pwd> @wfretry <item_type> <item_key>
```

Provide an item type and item key to uniquely identify an item or process instance. The script first returns the list of errored activities by label name. The script then prompts you for the label name of an activity that you wish to skip, retry, or reset. If you choose skip, then you must also specify the result that you want the skipped activity to have.



Attention: This script calls the WF_ENGINE HandleError API, so you can actually specify the label name of any activity

associated with the specified item type and item key to perform a rollback. See: `HandleError`: page 7 – 31.

Wfrmall.sql

Use `wfrmall.sql` to delete all data in all Oracle Workflow runtime and design time tables.

Use the script as follows:

```
sqlplus <user/pwd> @wfrmall
```



Warning: This script deletes **ALL** workflow definitions. Do not use this script unless you are absolutely sure you want to remove all workflow data from your runtime and design time tables.

Once you run this script, you should also reload the workflow definitions for the Standard, System:Mailer, and System:Error item types stored in the files `wfstd.wft`, `wfmail.wft`, and `wferror.wft`, respectively.

Wfrmitms.sql

Use `wfrmitms.sql` to delete status information in Oracle Workflow runtime tables for a particular item. This script prompts you to choose between deleting all data associated with a specified item type and item key or deleting only data for the completed activities of the specified item type and item key.

Use the script as follows:

```
sqlplus <user/pwd> @wfrmitms <item_type> <item_key>
```

Wfrmitt.sql

Use `wfrmitt.sql` to delete all data in all Oracle Workflow design time and runtime tables for a particular item type. This script prompts you for an item type from a list of valid item types.

Use the script as follows:

```
sqlplus <user/pwd> @wfrmitt
```



Warning: This script deletes **ALL** workflow data for a specified item type.

Wfrmtime.sql

Use `wfrmtime.sql` to delete runtime data associated with a given item type. This script prompts you for an item type to purge, from a list of valid item type, then asks you to choose between deleting all data associated with a specified item type or deleting only data for the completed activities and items of the specified item type.

Use the script as follows:

```
sqlplus <user/pwd> @wfrmtime
```

Wfrmita.sql

Use `wfrmita.sql` to delete all workflow data for a specified item type attribute. Currently you can use Oracle Workflow Builder to delete item type attributes saved in a file, but not in a database. If you need to delete item type attributes from a database, use this script instead. This script prompts you for the item type and the name of the attribute to delete.

Use the script as follows:

```
sqlplus <user/pwd> @wfrmita
```

Wfrun.sql

Use `wfrun.sql` to create and start a specified process.

Use the script as follows:

```
sqlplus <user/pwd> @wfrun <item_type> <item_key> <process_name>
```

Wfstatus.sql

Use wfstatus.sql to display an end user status report for an indicated item. The output is 132 characters per line.

Use the script as follows:

```
sqlplus <user/pwd> @wfstatus <item_type> <item_key>
```

Wfstat.sql

Use wfstat.sql to display a developer status report for an indicated item. The output is 132 characters per line.

Use the script as follows:

```
sqlplus <user/pwd> @wfstat <item_type> <item_key>
```

Wfver.sql

Use wfver.sql to display the version of the Oracle Workflow server, the status and version of the Oracle Workflow PL/SQL packages, and the version of the Oracle Workflow views installed.

Use the script as follows:

```
sqlplus <user/pwd> @wfver
```

Wfdirchk.sql

Use wfdirchk.sql to check for the following conditions in your directory service data model:

- Invalid internal names that contain the characters '#', ':', or '/' in WF_USERS.
- Invalid compound names in WF_USERS or WF_ROLES.
- Duplicate names in WF_USERS or WF_ROLES.
- Multiple names in WF_USERS or WF_ROLES linked to the same row in the original repository.
- Missing display names in WF_USERS or WF_ROLES.

- Invalid Notification Preference or null email address if the Notification Preference is MAILTEXT, MAILHTML, or SUMMARY in WF_USERS or WF_ROLES.
- Invalid Status in WF_USERS.
- Rows in WF_USERS that do not have a corresponding row in WF_ROLES.
- Invalid internal names in WF_ROLES that contain the characters '#' or '/' or have a length greater than 30 characters.
- Invalid user/role foreign key in WF_USER_ROLES.
- Missing user/role in WF_USER_ROLES (every user must participate in its own role).
- Duplicate rows in WF_USER_ROLES.

Wfdirchk.sql should return no rows to ensure that your directory service data model is correct.

Use the script as follows:

```
sqlplus <user/pwd> @wfdirchk
```

APPENDIX

A

Oracle Workflow Builder Menus and Toolbars

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

Oracle Workflow Builder Menus

The Oracle Workflow Builder main menu bar includes the following menus:

- File
- Edit
- View
- Window
- Help

File Menu

The File menu lets you perform several actions.

<u>N</u>ew	Creates a new workspace for you to define an item type.
O pen	Opens a data store by prompting you to connect to a database or a file.
<u>C</u>lose Store	Closes the selected data store. This menu option is available only if the Navigator is the active window.
<u>S</u>ave	Saves changes to the currently connected database or file.
S ave <u>A</u>s	Save changes to the file or database you specify with an optional effective date.
C reate S hortcut	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 95 or Windows NT 4.0 or higher. Earlier versions of Microsoft Windows NT do not support the concept of shortcuts.
<u>V</u>erify	Validates all process definitions in the current data store.

<u>P</u>rint Design	Prints the process diagram displayed in the active process window.
<u>S</u>how/<u>H</u>ide 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.
<u>L</u>oad <u>R</u>oles from <u>D</u>atabase	Loads the Oracle Workflow directory service roles from the current database store into Oracle Workflow Builder and makes them viewable from any property page poplist field that references roles. This menu option is available only if the current data store is a database.
<u>E</u>xit	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:

<u>N</u>ew	Creates a new item type, function activity, process activity, notification activity, message, lookup type, lookup code, or attribute by displaying its property page(s).
<u>C</u>opy	Copies the selected object in the navigator tree.
<u>P</u>aste	Pastes the object from the clipboard into the selected branch of the navigator tree.
<u>D</u>elete	Deletes the selected object from the navigator tree.
<u>F</u>ind	Displays the Search window so you can enter search criteria to find an object in the navigator tree.
<u>F</u>ind <u>A</u>gain	Finds an object in the navigator tree using the same criteria defined previously in the Search window.
<u>P</u>roperties	Shows the property pages of the selected object.
<u>P</u>rocess <u>D</u>etail	Opens the process window of the selected process activity.
<u>M</u>ove <u>A</u>tttribute	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:

<u>D</u>elete Selection	Deletes the selected object(s) from the process window.
<u>P</u>roperties	Shows the property pages of the selected activity node.
<u>C</u>opy Design	Copies the process diagram displayed in the active process window to the clipboard.

View Menu

The View menu lets you alter the display of Oracle Workflow Builder.

<u>F</u>ont	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.
<u>G</u>rid Snap	Toggles grid snap on or off for all process windows.
<u>L</u>og -> <u>S</u>how	Toggles between displaying and hiding the Log window. The Message Log window displays messages from the Workflow Builder that are not error-related.
<u>L</u>og -> <u>D</u>etailed	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.
<u>L</u>og -> <u>T</u>o <u>F</u>ile	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.
<u>L</u>og -> <u>B</u>ring to <u>F</u>ront	Brings the Message Log window to the front as the active window.
<u>S</u>how Label submenu	A submenu of options that let you control the information displayed in an activity's label. Choose either Display Name in Designer, Performer in Designer, or Comment in Designer.

Show Label -> Display Name in Designer	Uses the display name as the label for activities 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 -> Performer in Designer	Uses the activity performer as the label for activities 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 in Designer	Uses the activity's comment as the label for activities 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.
Show Label -> Internal Name in Navigator	Toggles between showing and hiding the internal name of all components in the Navigator window. The internal name, if shown, appears after the display name as (<INTERNAL_NAME>).

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.
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.

Window Menu

The Windows 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—Copy the selected object.



Paste—Paste the copied object into the current object branch.



Find—Display the Search window to specify the search criteria to locate an object in the navigator tree.



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:



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.



Find—Displays the process Overview window.



Help—Displays help on how to use Oracle Workflow.

APPENDIX

B

Oracle Applications Embedded Workflows

This appendix lists the workflows that are embedded in Oracle Applications and Oracle Self-Service Web Applications.

Predefined Workflows Embedded in Oracle Applications and Oracle Self-Service Web Applications

You can use Oracle Workflow to customize these predefined workflow processes. A full description of each workflow is documented in the respective product's User's Guide.

Application Implementation Wizard

Application Implementation Wizard provides a set of workflow processes that guide you through the setup and implementation of Oracle Applications. The Application Implementation Wizard helps you through the tasks and interdependencies of configuring Oracle Applications for your installation. To make your implementation job easier, the Application Implementation Wizard logically groups similar setup tasks.

The sequence of steps that the Wizard takes you through are contingent on the application modules you install. This obviates running duplicate setup steps when implementing multiple application modules.

The details and usage of the workflow processes can be obtained from the Application Implementation Wizard User's Guide.

Oracle Web Employees

Expense Reporting Workflow—Used by Oracle Web Employees and Oracle Payables to process both manager approval and accounting review of Web Employees–entered expense reports. The process controls if Web Employees–entered expense reports can be imported from the Invoice Import Interface tables into Oracle Payables invoice tables for payment. It also notifies employees at key events during the workflow.

Candidate Offer Approval Process—Submits an offer made using the Candidate Offers option in Line Manager Direct Access to the appropriate managers in the approval hierarchy. When the last approver in the hierarchy approves the offer, the process notifies Human Resources to print, sign and post the offer letter to the candidate and waits for the candidate's response. Once the candidate responds to the offer, the originating manager is notified and the process completes. The process keeps the originating manager informed of the offer status throughout the process.

Employee Direct Access—Includes processes that allow employees to view and update their personal details on the Web, including:

- Name
- Address
- Phone Numbers
- Contact Persons
- Marital Status
- Resume
- Qualifications
- Personal Competence Profile
- School and College Attendances
- Work Choices
- Succession Plan

Also includes processes to allow employees to enroll in a class and apply for a job. Three of the processes (Change Marital Status, Enroll in a Class, and Apply for a Job) include a standard approvals subprocess so that the relevant managers or Human Resources personnel can check the employee's entry before it is committed to the database. You can add this standard approvals subprocess to other processes, as required.

Person Search Process—Controls how a user navigates between entering search criteria and producing lists of people based on the criteria.

Person Suitability Match Process—Controls how a person navigates between entering matching criteria, showing list of people who match the criteria and displaying bar chart representations of their skills.

Career Management Reviews Process—Sends notifications to reviewers for Appraisals and Assessments.

360 Degree Assessment Process—Sends a notification to a set of people to indicate that they should perform an assessment as part of a group. Also handles the responses once the assessments are complete.

Receipt Confirmation Process—This process allows you to confirm receipts on the web.

Requisition Approval Process—Submits a requisition created from Web Requisitions to the appropriate managers for approval and updates the status of the requisition.

Oracle Web Customers

Customer Self-Service Registration Approval Process—Oracle Web Customers allows a guest to logon and register as a customer contact

for a company. This process routes a notification to the appropriate account approver, as defined by the Web Store owner, to verify and approve the registration. If the account approver approves the registration, the guest's account for both the Web User Logon and the Order Entry System Customer Account is activated. If the account approver rejects the registration, both accounts are deactivated. If the guest has been identified as a possible duplicate contact in the customer registry, the process allows the account administrator of the Web Store to associate the request with an existing contact, accept the request as a new contact registration or reject the request all together.

Order Entry Review Process—Oracle Web Customers allows you to set up a reviewer to verify order entry on the Web. This process routes an order notification to the reviewer for approval. If approved, the quote is booked and the order status is changed to 'Saved', otherwise, the order is rejected. In either case, appropriate notifications are sent to the customer or to the sales representative who prepared the order for the customer.

Oracle Web Suppliers

Supplier Self-Service Registration Approval Process—Oracle Web Suppliers allows a guest to logon 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 Engineering

Engineering Change Orders—Submits an engineering change order to the appropriate people for approval.

Oracle General Ledger

Journal Approval Process—Journals can now be approved before posting. Create an approval hierarchy and define authorization limits for each user. The Journal Approval process automatically routes journals to the appropriate user, based on the approval hierarchy.

Oracle Payables

AP Open Interface Import Process—Automates verification and validation of data in the Open Interface tables. For example, this process can be modified to validate all accounting code combinations

in the Open Interface 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.

Credit Card Transaction Employee Workflow—Notifies and confirms credit card transactions with a card holder. Oracle Payables initiates this process after you submit the Credit Card Transaction Validation and Exception Report. The process notifies an employee of transactions created by the employee's credit card, and confirms the transaction information with the employee.

Credit Card Transaction Manager Workflow—Notifies and confirms credit card transactions with a card holder's manager. Oracle Payables initiates this process when the Credit Card Transaction Employee Workflow executes. The Credit Card Transaction Manager Workflow notifies a manager of transactions created by the employee's credit card, and determines if the manager needs to approve the transactions.

Expense Reporting Workflow—Used by Oracle Web Employees and Oracle Payables to process both manager approval and accounting review of Web Employees-entered expense reports. The process controls if Web Employees-entered expense reports can be imported from the Invoice Import Interface tables into Oracle Payables invoice tables for payment. It also notifies employees at key events during the workflow.

Oracle Projects

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.

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.

Oracle Purchasing

Document Approval Process—Performs all approval related activities that existed in Oracle Purchasing. These include, but are not limited to, document submission, approval, forwarding, approval notifications, and reject.

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.

Change Orders Process—Part of the overall Procurement Workflow. This process allows you to control which changes require purchasing documents to be reapproved and which changes increment the document revision.

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.

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.

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.

Attribute See Activity Attribute, Item Type Attribute, or Message Attribute.

Background Engines A supplemental Workflow Engine that processes deferred or timed out activities.

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.

Directory Services A mapping of Oracle Workflow users and roles to a site's directory repository.

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.

Item A specific process, document, or transaction that is managed by a workflow process. For example, the item managed by the Requisition Approval Process workflow is a specific requisition created by Oracle Internet Commerce's Web Requisitions page.

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. For example, PO Requisition is an item type used to group all requisitions created by Oracle Internet Commerce's Web Requisitions page. 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 is 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 predefined 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 concurrent program that sends E-mail notifications to users via a mail application, and processes E-mail responses.

Notification Web Page A Web page that you can view from any Web browser 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.

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.

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

A

- AbortProcess(), 7 – 18
- Access Level, 2 – 49
 - default, 2 – 52
- Access level indicator, 3 – 24
- Access property page, 3 – 24
- Access protection
 - See also* Access level; Protection level
 - preserving customizations, 3 – 25
- AccessCheck(), 7 – 89
- ACCOUNT parameter, 2 – 29
- ACTID, 6 – 3, 6 – 7
- Activities, 3 – 8, 3 – 41
 - accessing from different data stores, 4 – 6, 5 – 2
 - copy, 3 – 51
 - cost, 3 – 43
 - create, 3 – 44, 3 – 46, 3 – 48
 - deferred, 3 – 43
 - effective date, 3 – 50
 - error process, 3 – 50
 - for an error process, 5 – 16
 - function, 3 – 41, 3 – 42
 - icons, 2 – 47, 3 – 45, 3 – 47, 3 – 49
 - in a loop, 3 – 50
 - in the Notify Approver subprocess, 10 – 16
 - in the Requisition Approval process, 10 – 10
 - joining branches, 4 – 4
 - notification, 3 – 41
 - optional details, 3 – 50
 - process, 3 – 41, 3 – 42
 - processing cost, 7 – 4
 - result type, 3 – 44, 3 – 47, 3 – 49
 - Standard, 3 – 41, 5 – 2
 - statuses, 7 – 3
 - System: Error, 3 – 41
 - timing out, 3 – 50
 - version number, 3 – 51
- Activities(), 7 – 45
- Activity attributes
 - See also* Function activity attributes
 - setting values for, 4 – 8
- Activity nodes
 - in the Notify Approver subprocess, 10 – 16
 - in the Requisition Approval process, 10 – 10
- AddAttr(), 7 – 81
- AddItemAttr(), 7 – 21
- Administrator privileges, 2 – 18
- And activity, 5 – 2
- APIs, 7 – 3
- Arrows, 4 – 2
- Assign activity, 5 – 15
- AssignActivity(), 7 – 30
- Attributes
 - copy, 3 – 23
 - type, 3 – 15, 3 – 19, 3 – 35, 3 – 40
- Automatic Notification Handler, 8 – 20
- Automatic responses, 8 – 20
- Automatic routing, 8 – 20

B

- Background engine, scripts, 11 – 3
- Background Engines
 - about, 2 – 43

- scripts, 11 – 3
- starting, 2 – 44
- submitting, 2 – 44
- Background(), 7 – 19
- BeginActivity(), 7 – 27
- Block activity, 5 – 4

C

- Callback functions, 6 – 5
 - command, 6 – 7
 - for item types, 3 – 17
- Cancel(), 7 – 75
- CancelGroup(), 7 – 76
- CLEAR(), 7 – 35
- Compare Date activity, 5 – 3
- Compare Number activity, 5 – 3
- Compare Text activity, 5 – 3
- Comparison activities, 5 – 3
- CompleteActivity(), 7 – 28
- Concurrent programs
 - Notification Mailer, 2 – 25, 2 – 26
 - Workflow Background Process, 2 – 44
 - Workflow Definitions Loader, 2 – 56
 - Workflow Resource Generator, 2 – 15
- CONNECT parameter, 2 – 29
- CONTEXT(), 7 – 39
- Continue Flow activity, 5 – 13
- Coordinating master/detail activities, 5 – 12
- Copy, limitation, 3 – 2
- Cost threshold, 3 – 43
- CreateProcess(), 7 – 9
- CurrentUser(), 7 – 54
- Custom logos, in the Notification Web page, 2 – 19
- Customization Level, 2 – 52
 - for activities, 3 – 18, 3 – 21, 3 – 27, 3 – 32, 3 – 45, 3 – 47, 3 – 49, 3 – 51

D

- DAD, 2 – 21

- Database Access Descriptor, 2 – 21
- Database Connection Descriptor, 2 – 21
- DCD, 2 – 21
- DEBUG parameter, 2 – 31
- Deferred activities, 2 – 43, 3 – 43
- Deferred processing, 2 – 43, 7 – 4
- Delete
 - all workflow data, 11 – 5
 - data for an item type, 11 – 5
 - item type attributes, 11 – 6
 - runtime data for an item type, 11 – 6
 - workflow status information, 11 – 5
- Detail process, 5 – 13
- Diagram arrows, 4 – 2
- Directory repository, 2 – 7
- Directory services, 2 – 7
 - checking the data model, 2 – 11, 11 – 7
 - integrating with local workflow users, 2 – 12
 - integrating with native Oracle users, 2 – 11
 - integrating with Oracle HR, 2 – 11
- Directory Services APIs, 7 – 49
- DISCARD parameter, 2 – 32
- Document integration, 3 – 16, 3 – 20, 3 – 35, 6 – 9
- Dynamic priority, 3 – 39

E

- E-mail
 - HTML attachments, 8 – 10
 - with HTML attachments, 8 – 6
- E-mail notifications, 1 – 4, 2 – 25
 - and HTML attachments, 2 – 2
 - example response template, 8 – 8
 - modifying mail templates, 2 – 36
 - requirements, 2 – 2
 - summaries, 8 – 18
 - templates for, 8 – 7
- Edit menu, A – 3
- Effective dates, 3 – 10, 3 – 12, 3 – 50, 7 – 6
- Effectivity, dates of, 3 – 6
- END activities, 4 – 4
- End Activity, 5 – 7
- Engine thresholds, 2 – 45

- Environment variables
 - WF_ACCESS_LEVEL, 2 – 49, 2 – 53
 - WF_RESOURCES, 2 – 14
- Error activities, 5 – 16
- Error handling, 7 – 31
- Error process, 3 – 50, 5 – 16
- Error Processing, 7 – 5
- Errored activities, retrying, 11 – 4
- Example function activity
 - Select Approver, 10 – 26
 - Verify Authority, 10 – 29
- Example process, Requisition Approval Process, 10 – 2

F

- FAILCOMMAND parameter, 2 – 31
- File menu, A – 2
- FND_FNDWFIAS, 9 – 2
- FND_FNDWFNOT, 8 – 2
- Fonts
 - modifying, 4 – 13
 - setting, 4 – 13
- Forward(), 7 – 74
- FROM parameter, 2 – 29
- FUNCMODE, 6 – 3, 6 – 4
- Function activities, 3 – 42
 - create, 3 – 46
 - standard PL/SQL API, 6 – 2
- Function activity attributes, 3 – 19, 3 – 47
- Functions, 3 – 8
 - See also* PL/SQL procedures

G

- Get Monitor URL activity, 5 – 15
- GET_ERROR(), 7 – 36
- GetActivityAttrDate(), 7 – 26
- GetActivityAttrInfo(), 7 – 25
- GetActivityAttrNumber(), 7 – 26
- GetActivityAttrText(), 7 – 26
- GetActivityLabel(), 7 – 12

- GetAttrDate(), 7 – 86
- GetAttrInfo(), 7 – 83
- GetAttrNumber(), 7 – 86
- GetAttrText(), 7 – 86
- GetBody(), 7 – 88
- GetDiagramURL(), 7 – 58
- GetEnvelopeURL(), 7 – 60
- GetInfo(), 7 – 84
- GetItemAttrDate(), 7 – 23
- GetItemAttrInfo(), 7 – 24
- GetItemAttrNumber(), 7 – 23
- GetItemAttrText(), 7 – 23
- GetRoleInfo(), 7 – 52
- GetRoleName(), 7 – 57
- GetRoleUsers(), 7 – 50
- GetSubject(), 7 – 87
- GetText(), 7 – 85
- GetUserName(), 7 – 56
- GetUserRoles(), 7 – 51
- Global variables, 3 – 14

H

- HandleError(), 7 – 31
- Hardware requirements, 2 – 2
- Help menu, A – 6
- Home page, 8 – 27
- HTMLAGENT parameter, 2 – 31
- HTMLDESC parameter, 2 – 31
- HTMLFILE parameter, 2 – 31
- HTMLTYPE parameter, 2 – 31

I

- Icons, 2 – 47
 - viewing, 3 – 45, 3 – 47, 3 – 49
- IDLE parameter, 2 – 30
- Initiating a workflow process, 10 – 18
- IsPerformer(), 7 – 53
- Item type attributes, 3 – 14, 3 – 18, 3 – 19, 7 – 6
 - Workflow Demonstration, 10 – 6
- Item types, 3 – 7, 3 – 14
 - callback function, 3 – 17

- context reset, 6 – 5
- copy, 3 – 22
- creation, 3 – 17
- loading, 3 – 8, 3 – 9
- saving, 3 – 8
- selector functions, 3 – 16, 6 – 5
- Standard, 5 – 2
- System: Error, 5 – 16
- System: Mailer, 2 – 36
- Workflow Demonstration, 10 – 6

Item_Activity_Statuses(), 7 – 43

Item_Notifications(), 7 – 47

ITEMKEY, 6 – 3, 6 – 7

Items(), 7 – 44

ITEMTYPE, 6 – 3, 6 – 6

J

- Java monitor tool, 9 – 3
- JavaScript, support in a Web browser, 2 – 2
- Joining activities, 4 – 4

L

- Load balancing, 5 – 7
- Loading item types, 3 – 9
- LOG parameter, 2 – 30
- Lookup codes, copy, 3 – 29
- Lookup types, 3 – 7, 3 – 26
 - copy, 3 – 29
 - creation, 3 – 27
- Loop Counter activity, 5 – 5
- Loop Reset, 4 – 3, 7 – 5
- Loops, 3 – 50, 6 – 4, 7 – 5

M

- Master process, 5 – 13
- Master/Detail coordination activities, 5 – 12
 - notes on usage, 5 – 14
- Menus, Oracle Workflow Builder, A – 2

- Message attributes, 3 – 30, 3 – 33, 3 – 34, 10 – 33
 - for Workflow Cancelled Mail message, 2 – 38
 - for Workflow Closed Mail message, 2 – 41
 - for Workflow Invalid Mail message, 2 – 39
 - for Workflow Open FYI Mail message, 2 – 38
 - for Workflow Open Mail message, 2 – 37
 - for Workflow Summary Mail message, 2 – 41
 - for Workflow Warning Mail message, 2 – 42
- Respond, 3 – 31, 3 – 36, 3 – 38
- RESULT, 10 – 33
- Send, 3 – 30, 3 – 36
- source, 3 – 30, 3 – 36

Message templates, for E-mail notifications, 2 – 36

Messages, 3 – 7

- body, 3 – 33, 10 – 32
- copy, 3 – 40
- creation, 3 – 31
- priority, 3 – 39
- subject, 3 – 33, 10 – 32
- viewing, 10 – 33

Messages window, 3 – 30

Monitoring

- Workflow Monitor, 9 – 3
- Workflow Status form, 9 – 2
- workitems, 1 – 4

Multilingual support, 11 – 2

N

- Naming conventions, PL/SQL stored procedures, 10 – 10
- Navigator Toolbar, A – 6
- Navigator tree, finding objects in, 3 – 5
- NODE parameter, 2 – 29
- Nodes
 - adding to a process, 4 – 5
 - start and end, 4 – 7
- NOOP activity, 5 – 5
- Notification, status, 11 – 4
- Notification access keys, 8 – 7
- Notification activities, 3 – 41
 - create, 3 – 44
 - dynamic priority, 3 – 39

- Notify Requisition Approval Required, 10 – 32
- Notification APIs, 7 – 67, 7 – 69
- Notification IDs, 8 – 7
- Notification Mailer
 - about, 2 – 25
 - configuration file, 2 – 28
 - response processing, 2 – 34
 - script to restart, 2 – 33
 - starting, 2 – 26
 - starting for MAPI-compliant applications, 2 – 27
 - starting for Oracle Office, 2 – 26
 - starting for UNIX Sendmail, 2 – 26
- Notification method, 8 – 2
- Notification summaries, via E-mail, 8 – 18
- Notification System, 2 – 25, 7 – 67
- Notification templates, for E-mail notifications, 2 – 36
- Notification Viewer form, requirements, 2 – 2
- Notification Web page, 1 – 4
 - reassigning notifications, 8 – 18
- Notifications, 8 – 2
 - dependence on directory services, 8 – 2
 - drill down to a form, 8 – 6
 - drill down to a URL reference, 8 – 11
 - identifying the responder, 7 – 77
 - load balancing, 5 – 7
 - reassign in Notification Viewer, 8 – 4
 - reassign in Notification Web page, 8 – 18
 - reassign via E-mail, 8 – 11
 - respond in Notification Viewer, 8 – 5
 - responding by E-mail, 8 – 7
 - responding by E-mail HTML attachment, 8 – 10
 - responding with Notification Web page, 8 – 18
 - via E-mail, 2 – 25, 8 – 6
 - via Notification Viewer, 8 – 3
 - via Notification Web page, 8 – 12
- Notifications(), 7 – 46
- Notify Approver, example notification activities, 10 – 32

- Notify Approver subprocess activities
 - End, 10 – 18
 - Notify Requisition Approval Required, 10 – 16
 - Or, 10 – 17
 - Reminder–Approval Needed, 10 – 17
 - Start, 10 – 16
- Notify Requisition Approval Required, 10 – 32

O

- OpenNotificationsExist(), 7 – 80
- Or activity, 5 – 2
- Oracle Office, 2 – 25
 - required folders, 2 – 32
- Oracle WebServer
 - identifying the workflow web agent, 2 – 15
 - modifying workflow web templates, 2 – 19
 - securing workflow web pages, 2 – 21
 - setting up the Workflow Monitor, 2 – 19
- Oracle Workflow, implementation issues, 2 – 4
- Oracle Workflow Builder, 1 – 3
 - Loader functionality, 3 – 11
 - overview, 3 – 2
 - requirements, 2 – 2
 - save modes, 3 – 11, 3 – 24
 - starting from command line, 3 – 13
- Oracle Workflow views, 7 – 62

P

- Personal Inbox, 1 – 4
 - See also* Notification Viewer form
- PL/SQL, 1 – 3
 - document, 6 – 9
- PL/SQL APIs
 - for a 'PL/SQL' document, 6 – 9
 - for a selector or callback function, 6 – 5
 - for function activities, 6 – 2
- PL/SQL stored procedures
 - creating, 10 – 10
 - naming conventions, 10 – 10
 - scripts, 10 – 10
- Preferred notification method, 8 – 2

- Preserving customizations, for an activity, 3 – 25
- Process activities, 3 – 42
 - create, 3 – 48
- Process diagram
 - adding nodes, 4 – 5
 - drawing, 4 – 2, 4 – 5
- Process rollback, 7 – 31
- Process window, 4 – 2
 - editing, 4 – 2
- Process Window Toolbar, A – 7
- Processes
 - activity transitions, 4 – 2
 - copying to clipboard, 4 – 12
 - creation, 3 – 6
 - editing, 3 – 8
 - loops, 6 – 4, 7 – 5
 - overview, 4 – 11
 - printing, 4 – 12
 - starting, 4 – 4
 - verify, 4 – 12
- Protection level, 2 – 50
 - reset, 11 – 4
- Protection level locking. *See* Access protection

R

- RAISE(), 7 – 38
- Reassign notifications
 - in Notification Viewer, 8 – 4
 - in Notification Web page, 8 – 18
 - via E-mail, 8 – 11
- REPLYTO parameter, 2 – 31
- Requirements, hardware and software, 2 – 2
- Requisition Approval Demonstration, web page, 10 – 18
- Requisition Approval Process, 10 – 2
 - example function activities, 10 – 26
 - installing, 10 – 3
- Requisition Approval process, initiating, 10 – 18

- Requisition Approval process activities
 - Approve Requisition, 10 – 14
 - End, 10 – 14
 - Notify Approver, 10 – 12
 - Notify Preparer No Approver Available, 10 – 11
 - Notify Requestor of Approval, 10 – 14
 - Notify Requestor of Forward, 10 – 11
 - Notify Requestor of Rejection, 10 – 13
 - Record Requisition Forward, 10 – 12
 - Reject Requisition, 10 – 13
 - Select Approver, 10 – 11
 - Start, 10 – 11
 - Verify Authority, 10 – 13
- Reset process. *See* Rollback
- RESPOND, 7 – 68
- Respond attributes, 2 – 37
- Respond to notification
 - in Notification Viewer, 8 – 5
 - via E-mail, 8 – 7
 - via Notification Web page, 8 – 12
- Respond(), 7 – 77
- Responder, 7 – 77
- Responder(), 7 – 78
- Response processing, by Notification Mailer, 2 – 34
- RESULT, 6 – 3, 6 – 7
- Result type, for activities, 3 – 44, 3 – 47, 3 – 49
- ResumeProcess(), 7 – 17
- Role
 - administrator, 2 – 18
 - property page, 4 – 18
- Role Resolution activity, 5 – 7
- Roles, 4 – 16
 - loading into the Workflow Builder, 4 – 16
 - tab page, 4 – 16
- Rollback, of process, 7 – 31
- Routing, automatic, 8 – 20
- Routing rules
 - deleting, 8 – 26
 - for a role, 8 – 21
 - listing, 8 – 20
 - overriding, 8 – 23
 - updating, 8 – 26

S

- Select Approver function activity, 10 – 26
- Selector functions, 3 – 16, 6 – 5
- SEND, 7 – 67
- Send(), 7 – 69
- SendGroup(), 7 – 72
- SetAttrDate(), 7 – 82
- SetAttrNumber(), 7 – 82
- SetAttrText(), 7 – 82
- SetItemAttrDate(), 7 – 22
- SetItemAttrNumber(), 7 – 22
- SetItemAttrText(), 7 – 22
- SetItemOwner(), 7 – 13
- SetItemParent API, 5 – 13
- SetItemParent(), 7 – 33
- SetItemUserKey(), 7 – 11
- Shortcuts, 4 – 14
- Shutdown files, 2 – 30
- SHUTDOWN parameter, 2 – 30
- Software requirements, 2 – 2
- SQL*Net, 2 – 2
- Standard activities, 5 – 2
- Standard APIs
 - for "PL/SQL documents", 6 – 9
 - for function activities, 6 – 2
 - for selector/callback functions, 6 – 5
- Standard item type, 5 – 2
- START activities, 4 – 4
- Start activity, 5 – 7
- StartProcess function, for demonstration, 10 – 22
- StartProcess(), 7 – 14
- Status report
 - developer, 11 – 7
 - end user, 11 – 7
- SUMMARYONLY parameter, 2 – 29
- SuspendProcess(), 7 – 16
- System: Error item type, 5 – 16
- System: Mailer item type, 2 – 36

T

- Tag files, 2 – 32
- TAGFILE parameter, 2 – 32
- TCP/IP drivers, 2 – 2
- TEST_ADDRESS parameter, 2 – 31
- Timed out processes, 2 – 43
- Timeout transitions, 4 – 3
- Timeouts, 3 – 50
- TOKEN(), 7 – 37
- Toolbars, Oracle Workflow Builder, A – 6
- Total(), 7 – 48
- Transitions, 4 – 2
 - creating, 4 – 10
 - editing, 4 – 10
- TRANSLATE(), 7 – 41
- Translation, 2 – 13

U

- UNIX Sendmail, 2 – 25
- UNPROCESS parameter, 2 – 32
- URLs
 - for Find Notifications Routing Rules web page, 8 – 21
 - for Find Notifications web page, 8 – 12
 - for Find Processes web page, 9 – 9
 - for Notifications Routing Rules web page, 8 – 20
 - for Oracle Workflow home page, 8 – 27
 - for Requisition Approval Demonstration web page, 10 – 19
 - for the Workflow Monitor, 9 – 8
 - for Worklist web page, 8 – 12
- UserActive(), 7 – 55

V

- Vacation forwarding, 8 – 20
- Verify Authority function activity, 10 – 29
- Version, 2 – 6, 7 – 6, 11 – 7
- Version number, for activities, 3 – 51
- Versioning, 3 – 6

- View menu, A – 4
- View notifications
 - E-mail summary, 8 – 18
 - electronic mail, 8 – 6
 - Notification Viewer, 8 – 3
 - Notification Web page, 8 – 12
 - web browser, 8 – 12
- Views, Oracle Workflow, 7 – 62
- VoteCount(), 7 – 79
- Voting activity, 5 – 8

W

- Wait activity, 5 – 3
- Wait for Flow activity, 5 – 13
- Web agent, for Oracle Workflow, 2 – 15
- Web home page, 8 – 27
- Web notifications, requirements, 2 – 3
- WF_ACCESS_LEVEL, 2 – 49, 2 – 53
- WF_ADMIN_ROLE, 2 – 18
- WF_ENGINE.AbortProcess, 6 – 4
- WF_ENGINE.BACKGROUND, 2 – 44
- WF_ITEM_ACTIVITY_STATUSES_V, 7 – 62
- WF_LANGUAGES view, 2 – 13
- WF_NOTIFICATION_ATTR_RESP_V, 7 – 64
- WF_REQDEMO.SelectApprover, 10 – 11, 10 – 26
- WF_REQDEMO.StartProcess, 10 – 18
- WF_REQDEMO.VerifyAuthority, 10 – 13, 10 – 29
- WF_RESOURCES, environment variable, 2 – 14
- WF_ROLES, view, 2 – 9
- WF_STANDARD.NOOP, 10 – 11, 10 – 16, 10 – 18
- WF_USER_ROLES, view, 2 – 10
- WF_USERS, view, 2 – 7
- WF_WEB_AGENT, 2 – 15
- Wfbkg.sql, 11 – 3
- Wfbkgchk.sql, 11 – 3
- Wfdirchk.sql, 11 – 7
- wfmail.cfg, 2 – 28

- WFNLADD.sql, 11 – 2
- Wfntfsh.sql, 11 – 4
- Wfprot.sql, 11 – 4
- Wfretry.sql, 11 – 4
- Wfrmall.sql, 11 – 5
- Wfrmita.sql, 11 – 6
- Wfrmitms.sql, 11 – 5
- Wfrmitt.sql, 11 – 5
- Wfrmtime.sql, 11 – 6
- Wfrun.sql, 11 – 6
- WFRUNDEMO.SQL, 10 – 18
- Wfstat.sql, 11 – 7
- Wfstatus.sql, 11 – 7
- Wfver.sql, 11 – 7
- Windows menu, A – 5
- WorkCount(), 7 – 90
- Workflow, sample, 10 – 3
- Workflow administrator, 2 – 18
- Workflow Builder menus, A – 2
- Workflow Cancelled Mail message template, 2 – 38
- Workflow Closed Mail message template, 2 – 41
- Workflow definitions
 - loading, 1 – 3
 - source control, 3 – 8
 - transferring, 2 – 54
- Workflow Definitions Loader, 1 – 3, 2 – 54
 - concurrent program, 2 – 56
- Workflow Designer. *See* Oracle Workflow Builder
- Workflow Engine, 1 – 3
 - calling after activity completion, 7 – 4
 - calling for activity initiation, 7 – 3
 - CANCEL mode, 7 – 6
 - core APIs, 7 – 34, 7 – 42
 - cost threshold, 3 – 43
 - deferred activities, 7 – 4
 - directory services, 7 – 49
 - error processing, 7 – 5
 - looping, 7 – 5
 - master/detail processes, 7 – 33
 - requirements, 2 – 2
 - RUN mode, 7 – 6

- threshold cost, 2 – 45, 7 – 4
- Workflow Engine APIs, 7 – 3, 7 – 8
- Workflow Invalid Mail message template, 2 – 39
- Workflow Monitor, 9 – 3
 - Administration buttons, 9 – 7
 - Detail Tab window, 9 – 5
 - Process Diagram window, 9 – 4
 - Process title, 9 – 4
 - setup, 2 – 19
- Workflow Open Mail message template, 2 – 36, 2 – 37
- Workflow processes
 - creating and starting, 11 – 6
 - monitoring, 9 – 3
- Workflow Report API, 7 – 58
- Workflow Resource Generator, 2 – 15
 - concurrent program, 2 – 16
- Workflow roles, 2 – 7
- Workflow Status form, 9 – 2
- Workflow Summary Mail message template, 2 – 41
- Workflow users, 2 – 7
- Workflow Warning Mail message template, 2 – 42
- Workflow web pages, modifying template, 2 – 19
- Workitems. *See* Items

Reader's Comment Form

Oracle Workflow Guide A56104-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information we use 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? What did you like least about it?

If you find any errors or have any other suggestions for improvement, please indicate the topic, chapter, and page number below:

Please send your comments to:

Oracle Applications Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065 USA
Phone: (650) 506-7000 Fax: (650) 506-7200

If you would like a reply, please give your name, address, and telephone number below:

Thank you for helping us improve our documentation.